Systemliteratur

NIXDORF
COMPUTER

Vorverarbeitungssystem 620/45

Programmierhandbuch

Produkt-/ Software-Marketing

05009.29.4.93

NIXDORF COMPUTER

Systemliteratur

Vorverarbeitungssystem Nixdorf 620

System-Software

Programmierhandbuch

Software-Versionen C und D

Änderung 1 vom 1. 10. 78

Bestell-Nr.: 29013.01.1.93

Ihre Aufnahme in die Verteilerliste für den Änderungsdienst erfolgt nur, wenn Sie diese Karte einschicken

Bitte senden Sie Änderungen und Ergänzungen zu diesem Literaturteil an die folgende Anschrift:

Name: _____

in Firma: _____

Straße: _____

Ort: _____

Land: _____

Zuständige NIXDORF-Niederlassung (unbedingt angeben):

Verkehrsnummer:
05009.29.4.93

Ausgabedatum der letzten Änderung lt. Organisationsblatt (unbedingt angeben):

.....

Postkarte

NIXDORF COMPUTER AG
Abt. VP 03 - ZSI
Fürstenallee 7

D-4790 PADERBORN

Bitte ausschneiden und in die Tasche im Handbuchrücken einstecken.

Systemliteratur
620/45

_____

_____

_____

_____

	Änderungswünsche/Fehler

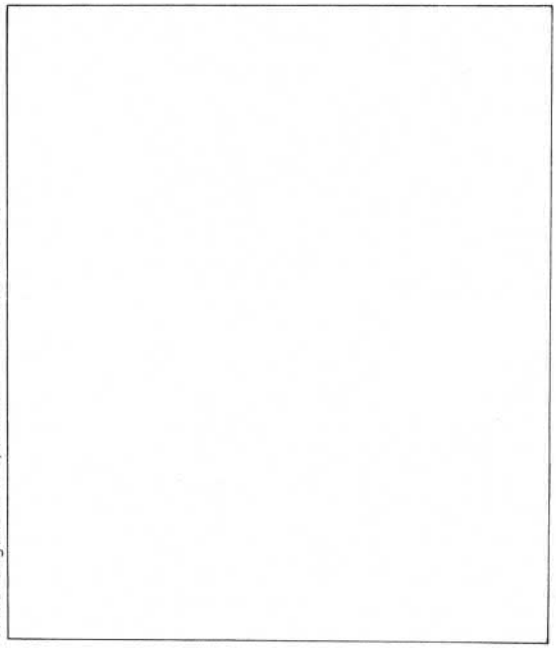
Änderungswünsche/Fehler

Sollten Ihnen bei der Benutzung dieses Teils der Systemliteratur Fehler aufgefallen sein, oder sollten Sie Vorschläge zur Verbesserung des Moduls haben, so bitten wir Sie, diese auf dem unten aufgedruckten Formular festzuhalten und an folgende Anschrift zu schicken.

NIXDORF COMPUTER AG
Abt.: VP 03 - ZSI
Fürstenallee 7
479 Paderborn

NIXDORF COMPUTER AG
Abt.: VP 03 - ZSI
Fürstenallee 7
479 Paderborn

Änderungswünsche/Fehler Verkehrsnr.:



Organisationsblatt

Organisationsblatt

Dieses Blatt gibt eine Übersicht über alle Änderungen, die seit der ersten Auflage vom 01.02.78 an diesem Modul durchgeführt wurden. Es wird bei jeder Änderungsmitteilung mitgeliefert und ist jeweils auszutauschen.

Erstauflage:	01.02.78	gültig für SDP-2
Änderung 1:	01.10.78	gültig ab SDP-3

Die Änderung umfaßt folgende Seiten:

3-6, 15, 27, 55, 79, 87-90, 90/1, 91, 100, 100/1, 127, 141, 143, 155, 159, 178-181, 183-183/1, 185, 189, 193, 197, 198, 201-201/1, 205, 209, 213, 215, 223, 225-226, 229-236/11, 268-268/5, 274-275.

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Datenorganisation

1. Datenorganisation

Alle in das System 620/45 eingegebenen Daten werden in Magnetplattendateien gespeichert. In den folgenden Abschnitten wird die Struktur und Organisation dieser Dateien beschrieben.

1.1 Feld

Die kleinste Organisationseinheit zur Aufnahme von Daten ist das Feld. Der Inhalt eines Feldes ist eindeutig definiert, z.B. Artikelnummer, Kundennummer oder Kundenname.

1.2 Satz

Der Satz ist die Zusammenfassung mehrerer Felder zu einer logischen Einheit. So wird man z.B. aus allen Feldern, welche die Daten eines Kunden enthalten, einen Kundenstammsatz bilden. Jeder Satz muß mit einem Eingabeformat beschrieben werden.

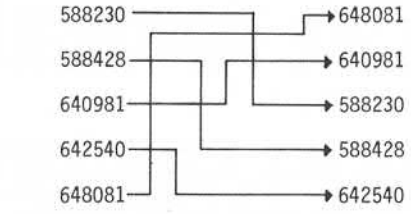
1.3 Stapel

Der Stapel wird aus einer Anzahl zusammengehörender Sätze gebildet. Ein Stapel kann entweder von einer Erfassungskraft per Tastatur, oder durch ein anderes Eingabegerät (Band, Lochkarte, DFÜ) in das System eingegeben werden. Jeder Stapel enthält einen Namen, der nur einmal vergeben werden darf. Der Name besteht aus max. 10 Zeichen, wobei das erste Zeichen ein Buchstabe sein muß.

	Datenorganisation

1.4 Sortierter Stapel

Bestehende Stapel können nach frei wählbaren Kriterien beliebig oft sortiert werden. Der Stapel wird bei der Sortierung nicht verändert, es wird nur eine Tabelle gebildet, welche die Sortierbegriffe und Verweise auf die zugehörigen Sätze des Stapels enthält.



sortierte Tabelle mit Verweisen — unsortierter Stapel

1.5 Tabelle

Durch Aufruf einer Supervisorfunktion wird ein bestehender Stapel in eine Tabelle umgewandelt. Während der Datenerfassung können eingegebene Werte mit Tabellenwerten verglichen und/oder Tabellenwerte in den Eingabesatz übernommen werden. Die Methode der Tabellenverarbeitung wird durch das Eingabeformat bestimmt.

1.6 Indexdatei

Ein Stapel kann vom Anwenderprogramm nur sequentiell gelesen werden. Um den wahlfreien Zugriff auf die Sätze eines Stapels zu ermöglichen, muß der Stapel in eine Indexdatei umgewandelt werden. Auf die Sätze einer Indexdatei kann sowohl indiziert als auch sequentiell zugegriffen werden. Weiterhin ist es möglich, die gelesenen Sätze zu verändern und auf ihre ursprüngliche Position zurückzuschreiben.

Eingabeformat

2. Eingabeformat

Ein charakteristisches Merkmal des Systems 620/45 ist, daß die Daten in ihrer Struktur und Speicherungsform nur einmal für alle Anwendungen beschrieben werden müssen. Zur Beschreibung der Struktur eines Datensatzes dient das Eingabeformat. Ein Eingabeformat ist eine Folge von Parametern, die sowohl die Reihenfolge und den Aufbau der Eingabedaten als auch Prüfungen und automatische Funktionen, die während der Datenerfassung durchgeführt werden sollen, festlegt. Darüber hinaus kann von Anwenderprogrammen auf jedes Feld eines Datensatzes einfach durch Angabe der im Eingabeformat festgelegten Feldnummer zugegriffen werden. Die Tatsache, daß die verwendeten Datenfelder nicht in jedem Programm neu deklariert werden müssen, bedeutet für den Programmierer des Systems eine wesentliche Arbeitserleichterung.

2.1 Strukturierung der Daten

Der erste Schritt bei der Erstellung eines Eingabeformates ist die Gruppierung von logisch zusammengehörigen Feldern eines Ursprungsbeleges zu einem oder mehreren Datensätzen und der Entwurf des Bildschirmformates. Ein Satz entspricht in der Regel einem Bildschirminhalt und enthält die Daten eines Ursprungsbeleges. Erfordert die Darstellung eines Ursprungsbeleges mehr als einen Bildschirminhalt, so gibt es zwei Möglichkeiten:

- Der Ursprungsbeleg wird in mehrere Teile zerlegt. Jeder Teil des Beleges, der bis zu 10 Zeilen (bzw. 22 Zeilen) des Bildschirmes belegt, wird von einem separaten Eingabeformat beschrieben. Die einzelnen Eingabeformate werden so untereinander verknüpft, daß sich eine logische Reihenfolge der Felder des Ursprungbeleges ergibt. Es findet lediglich bei Wechsel des Eingabeformates ein Bildschirmwechsel statt. Dementsprechend besteht ein erfaßter Ursprungsbeleg aus mehreren Datensätzen.

- Der Ursprungsbeleg wird nur mit einem Eingabeformat beschrieben. Bei der Dateneingabe verschiebt das System - beginnend mit Zeile 11 bzw. 23 - den Bildschirminhalt zeilenweise nach oben, d.h. Zeile 1 des aktuellen Bildschirminhaltes wird hinausgeschoben und in der freiwerdenden Zeile 10 bzw. 22 kann die nächste Dateneingabe erfolgen. Dieses Verfahren nennt man "ROLL-UP-Verfahren".

	Eingabeformat

2.2 Entwurf des Bildschirmformates

Beim Entwurf des Bildschirmformates ist zwischen dem 480-Zeichen Bildschirm (10 Zeilen à 40 Zeichen Anwenderbereich) und dem 1920-Zeichen Bildschirm (22 Zeilen à 80 Zeichen Anwenderbereich) zu unterscheiden. Dementsprechend stehen unterschiedliche Codierformulare zur Verfügung. Auf dem Formular "DISPLAY-AUFBAU" werden die Daten anhand des Ursprungsbeleges in bezug auf Reihenfolge, Länge und Maskenangaben so dargestellt, wie sie später auf dem Bildschirm erscheinen sollen. Eine Zeile des Bildschirmaufbaus kann sowohl für Masken und Datenfelder, als auch für reine Masken verwendet werden. Masken haben die Funktion, die Eingabefelder zu benennen.

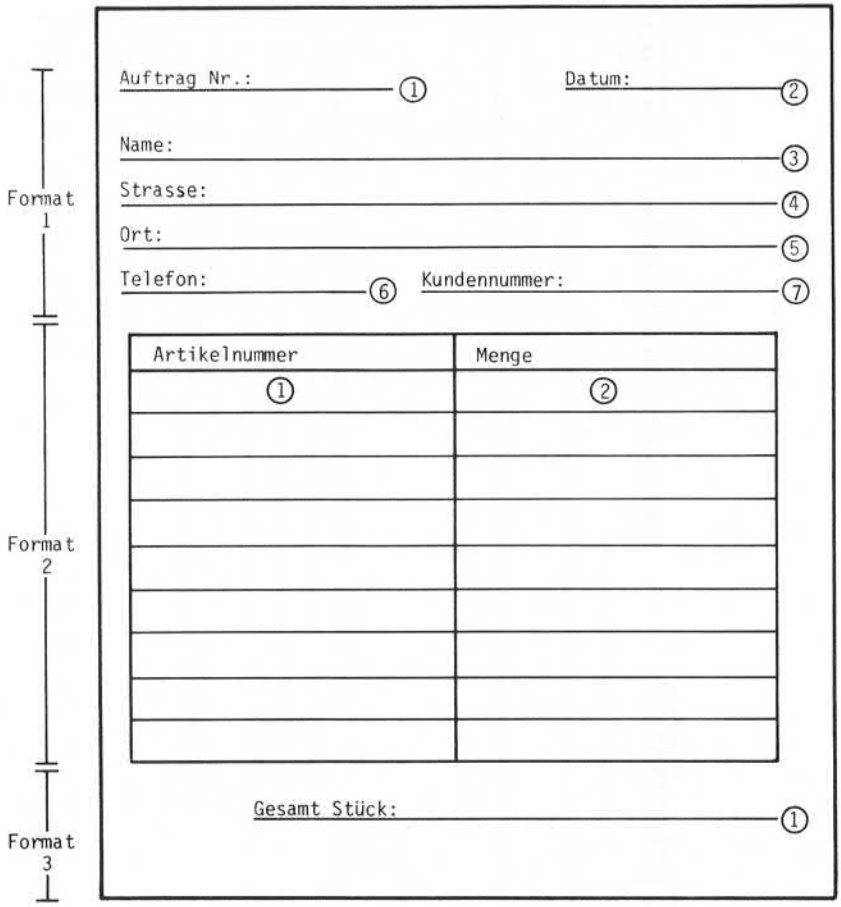
Die Länge der Masken ist auf max. 40 Zeichen begrenzt. Datenfelder können bis zu 99 Stellen lang sein und werden im Bildschirmformat durch Unterstriche dargestellt.

Folgende Punkte sollten beim Entwurf eines Bildschirmformates beachtet werden:

- Wählen Sie möglichst kurze Masken, da dadurch die Arbeitsgeschwindigkeit des Platzes positiv beeinflußt wird.
- Halten Sie sich beim Entwurf des Bildschirmformates möglichst an den Aufbau des Ursprungsbeleges.
- Sorgen Sie dafür, daß Masken und Eingabedaten klar zu unterscheiden sind. (Abgrenzen der Masken durch Klammern oder Doppelpunkt etc.).

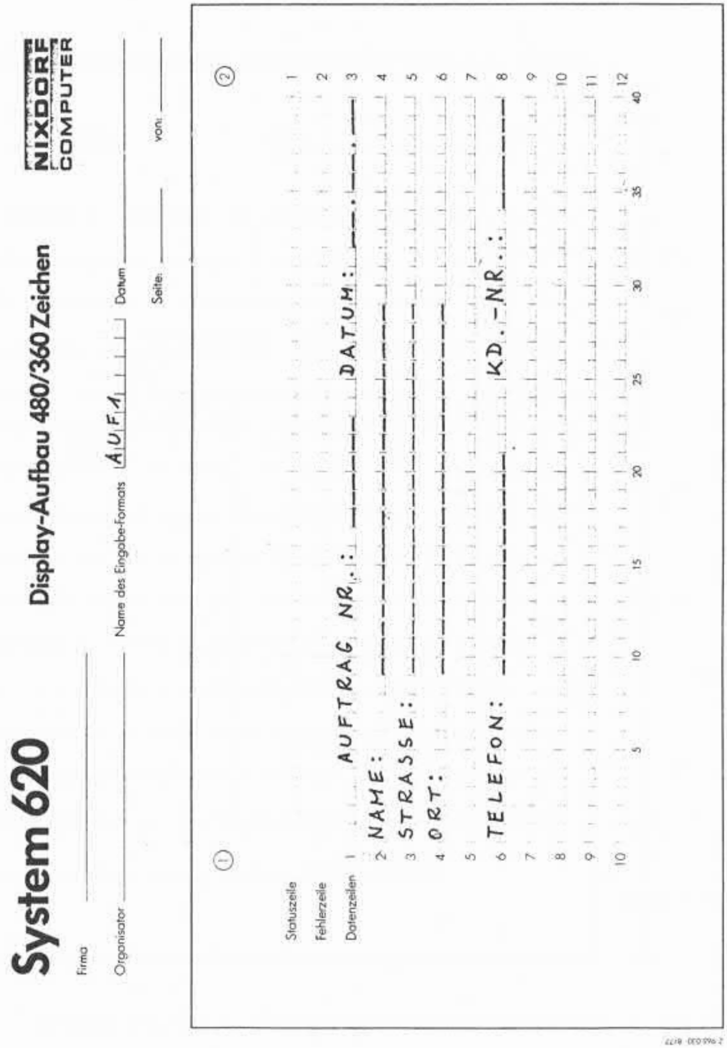
Auf der folgenden Seiten finden Sie ein Beispiel für einen Ursprungsbeleg mit daraus entwickelten Bildschirmformaten.

Eingabeformat



Ursprungsbeleg Auftragserfassung

	Eingabeformat



System 620

Display-Aufbau 480/360 Zeichen

NIXDORF COMPUTER

Firma _____

Organisator _____ Name des Eingabe-Formats AUF2 Datum _____

Seite: _____ von: _____

① ②

Statuszeile			1
Fehlerzeile			2
Datenzeilen	1	ARTIKELNUMMER , MENGE	3
	2	------ -----	4
	3		5
	4	BEI AUFTRAGSENDE "3" EINGEBEN: _	6
	5		7
	6		8
	7		9
	8		10
	9		11
	10		12

5 10 15 20 25 30 35 40

Z 960.530 8/77

① Zeilennummer f. Eingabeformat ② Zeilennummer f. Editier-Befehle wie: ACCEPT/PAUSE/SHOW

System 620

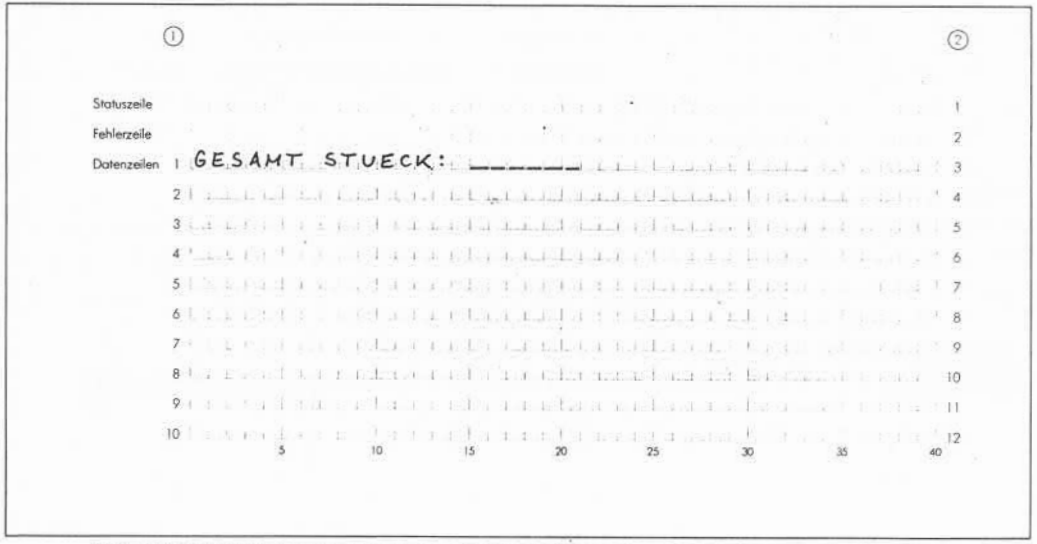
Display-Aufbau 480/360 Zeichen

NIXDORF COMPUTER

Firma _____

Organisator _____ Name des Eingabe-Formats AUF3 Datum _____

Seite: _____ von: _____



Z 960.070 8/77

① = Zeilennummer f. Eingabeformat · ② = Zeilennummer f. Editor-Befehle wie ACCEPT/PAUSE/SHOW

	Eingabeformat

2.3 Codierung des Eingabeformates

Das Eingabeformat setzt sich aus Parameterangaben zusammen, die auf dem Formular "EINGABE-FORMAT" codiert werden.

Jedes Eingabeformat erhält einen Namen, der nur einmal vergeben werden darf. Der Name besteht aus max. 8 Zeichen, wobei das erste Zeichen ein Buchstabe sein muß. Das Eingabeformat wird unter diesem Namen in der Eingabeformat-Bibliothek gespeichert und ist jederzeit abrufbar. Eine Zeile des Eingabeformat-Formulars beschreibt ein Feld des Datensatzes. Jede Zeile besteht aus 33 Parameterangaben. Nach den Vorgaben des Bildschirmformates und den Erfordernissen des Ursprungsbeleges werden die notwendigen Parameter codiert.

2.3.1 Feldnummer (Parameter 1)

Jedem Datenfeld innerhalb eines Eingabeformates muß eine Feldnummer zugeordnet werden. Durch diese Feldnummer wird das Datenfeld eindeutig identifiziert und kann durch Anwenderprogramme angesprochen werden. Eine Ausnahme bilden reine Maskenfelder, die keine Daten aufnehmen (z.B. Oberschriftszeilen). Diese Maskenfelder erhalten die Feldnummer "0". Die höchste mögliche Feldnummer ist 999, d.h. in einem Datensatz können max. 999 Felder enthalten sein. Werden nachträglich Felder eingefügt oder gelöscht, vergibt das System automatisch neue fortlaufende Feldnummern. Das gilt auch, wenn Feldnummern nicht in aufsteigender Folge eingegeben werden.

2.3.2 Zeile (Parameter 2)

Dieser Parameter legt die Bildschirmzeile fest, auf der die Maske und/oder die Eingabedaten angezeigt werden.

Soll ein Beleg im ROLL-UP-Verfahren eingegeben werden (siehe Punkt 2.1), wird beim Oberschreiten der letzten Bildschirmzeile mit Zeilennummer 11 bzw. 23 fortgefahren.

Wird die Zeilen- und Spaltennummer mit der FELD-AUSL-Taste übergangen, d.h. auf Null gesetzt, gilt das Feld als Dunkelfeld (siehe Punkt 2.3.7).

2.3.3 Spalte (Parameter 3)

Die Zeichenposition innerhalb der Bildschirmzeile, auf der die Maske und/oder die Eingabedaten angezeigt werden, legt dieser Parameter fest.

	Eingabeformat

2.3.4 Maske (Parameter 4)

Masken dienen zur Identifizierung von Datenfeldern oder zur Erläuterung von Erfassungsfunktionen für die Bedienungskraft. Eine Maske kann aus max. 40 Zeichen bestehen und wird mit dem Ungleich-Zeichen (≠) abgeschlossen. Das Ungleich-Zeichen erscheint nicht auf dem Bildschirm. Es legt lediglich fest, ab welcher Position die Eingabedaten angezeigt werden. Zwischen Maske und Ungleich-Zeichen sollte eine Stelle freigelassen werden, da sich anderenfalls die Daten lückenlos an die Maske anschließen. Eine Maske kann aus allen verfügbaren Tastaturzeichen bestehen. Besteht eine Maske aus mehr als 20 Zeichen, wird sie in der nächsten Zeile des Formulars forgesetzt. Soll ein Eingabefeld ohne Maske angezeigt werden, wird statt der Maske das Ungleich-Zeichen auf der ersten Position des Masken-Parameters codiert. Das Feld beginnt dann auf der durch Zeile und Spalte festgelegten Position.

Wird nur die Einblendung einer Maske ohne Eingabefeld gewünscht, wird im Parameter Feldnummer ein Null codiert. Das Ungleich-Zeichen als Begrenzung der Maske kann dann entfallen.

2.3.5 Feldlänge (Parameter 5)

Durch diesen Parameter wird die maximale Länge des Feldes festgelegt. Das Maximum sind 99 Zeichen. Für eine reine Maskeneinblendung wird als Feldlänge "0" codiert.

2.3.6 Feldtyp (Parameter 6)

Der Feldtyp bestimmt die Art der Daten, die in dieses Feld eingegeben werden können. Der Versuch, Zeichen einzugeben, die nicht dem Feldtyp entsprechen, wird vom System mit einer Fehlermeldung quittiert.

	Eingabeformat

Folgende Feldtypen sind möglich:

A Alpha-Feld. Alle Zeichen außer den Ziffern 0-9 sind erlaubt (unabhängig von der Art der Tastatur).

L Automatisches Umschalten in die untere (lower) Ebene, sowohl bei der Locher- als auch bei der Schreibmaschinentastatur. Ein Umschalten in die obere Ebene muß manuell vorgenommen werden. Alle Zeichen sind erlaubt.

M Automatisches Umschalten in die numerische Ebene bei beiden Tastaturarten. Ein manuelles Umschalten in die Alpha-Ebene ist möglich.

N Nur numerische Zeichen erlaubt.

U Automatisches Umschalten in die obere (upper) Ebene auf beiden Tastaturen.

- Auf der Locher-Tastatur ist mit Hilfe der ALPHA-Taste ein Umschalten in die untere Ebene möglich.
- Auf der Schreibmaschinen-Tastatur ist ein manuelles Umschalten in die untere Ebene nicht möglich. Aus diesem Grund wird bei Schreibmaschinen-Tastaturen anstelle von "U" vorzugsweise "L" oder "M" spezifiziert.

Wird keine Angabe über den Feldtyp gemacht, setzt das System automatisch "L" ein.

Bei der Auswahl eines geeigneten Feldtyps beachten Sie bitte die folgenden Punkte:

- Alpha (A) ist sinnvoll bei Feldern, die keine numerischen Zeichen enthalten können, wie z.B. Namen.
- Umschalten in die untere Ebene (L) sollte eingesetzt werden bei Feldern, die im wesentlichen aus Alpha-Zeichen bestehen, wie z.B. Straße, Ort.
- Umschalten in die numerische Ebene (M) sollte bei gemischter Bestückung des Systems mit Locher- und Schreibmaschinentastatur und überwiegend numerischen Daten verwendet werden.

	Eingabeformat

- Nur numerische Zeichen (N) können bei rein numerischen Feldern eingesetzt werden.

- Umschalten in die obere Ebene (U) darf nur bei Lochertastaturen zur Anwendung kommen.

2.3.7 Nicht anzeigbare (dunkelgesteuerte Felder)

Dunkelfelder sind Felder, die physikalisch im Stapel existieren, jedoch in keinem Modus angezeigt werden.

Ein Dunkelfeld wird wie ein normales Eingabefeld im Eingabeformat unter Angabe der Feldnummer und der möglichen bzw. erforderlichen Prüfungen definiert, mit dem Unterschied, daß sowohl Zeilen- als auch Spalten-Nr. mit der FELD-AUSL-Taste übergangen, d.h. auf Null gesetzt werden.

Während der Erfassung wird das Dunkelfeld wie ein Sprungfeld behandelt, d.h. der Cursor positioniert nicht in dieses Feld. Auch nach Herausnahme der AUTO-Taste können keine Eingaben erfolgen. Der Inhalt des Dunkelfeldes kann ausschließlich durch

- automatische Funktionen (Duplizieren, Erhöhen)
- Übernahme eines Tabellenwertes
- einen MOVE-Befehl des Anwenderprogrammes

verändert werden.

Eine Veränderung des Feldinhaltes kann weder durch KORR, FELD-KORR oder SATZ-KORR erfolgen. Eine Anzeige des Feldinhaltes ist nur mit Hilfe des SHOW-, PAUSE- oder OUTPUT-Befehls möglich.

Bemerkung:

Eine im Eingabeformat für ein Dunkelfeld angegebene Maske wird nicht auf dem Bildschirm angezeigt.

Wird durch eine der Feldprüfungen ein Fehler festgestellt, erscheint zwar die entsprechende Fehlermeldung, aber der Cursor positioniert auf das nächste Feld.

	Eingabeformat

Wird ein Dunkelfeld im Feldende-Editor durch einen POSITION-Befehl angesprochen, werden alle für dieses Feld spezifizierten Feldende-Prüfungen durchlaufen, und der Cursor positioniert auf das nächste Feld.

In der Statuszeile wird jeweils die physikalisch richtige Feldnummer angezeigt, d.h. die Dunkelfelder werden im Feldzähler berücksichtigt.

Ist ein Dunkelfeld das 1. Feld eines Satzes, ist ein Zurücksetzen des Cursors in den vorhergehenden Satz über die Tastenkombination SATZ- ← nicht möglich. In den vorhergehenden Satz kann nur durch eine Suchoperation zurückpositioniert werden.

2.3.8 Prüfen (Parameter 7)

Dieser Parameter bestimmt die Art der Datenprüfung im Prüf.-Modus. Die Dateneingabe wird nicht beeinflußt. Folgende Angaben sind möglich:

C Abhängigkeitsprüfung:

- Tritt bei Stapelkontrollrechnung mit diesem Feld eine Stapelsummendifferenz auf, müssen im PRÜF-Modus die Daten durch Neueingabe geprüft werden.
- Tritt bei Stapelkontrollrechnung mit diesem Feld keine Stapelsummendifferenz auf, erfolgt keine Prüfung der Daten.

E Eingabe

Der Eingabe-Modus wird simuliert und das Feld kann neu eingegeben werden.

K Prüfen durch Neueingabe:

Die im EINAGBE-Modus erfaßten Daten des mit "K" spezifizierten Feldes müssen im PRÜF-Modus erneut eingegeben werden. Der Inhalt des zu prüfenden Feldes wird nicht auf dem Bildschirm angezeigt. Das System vergleicht die Daten, die die Bedienungskraft im PRÜF-Modus eingibt, zeichenweise mit den Daten, die von der Bedienungskraft im EINGABE-Modus eingegeben wurden. Wird eine Ungleichheit festgestellt, erfolgt eine Fehlermeldung. Die Prüferin hat dann die Möglichkeit der Korrektur.

	Eingabeformat

R Ende des Prüfens für den derzeitigen Eingabesatz

Ab dieser Stelle werden alle nachfolgenden Felder des derzeitigen Eingabesatzes nicht mehr geprüft, einschließlich des Feldes für das "R" spezifiziert wurde.
Durch Benutzung dieser Funktion wird der Prüflauf beschleunigt.

S Bei Angabe von "S" erfolgt keine Prüfung der Daten durch Neueingabe. Masken und Daten erscheinen nur zur Sichtprüfung auf dem Bildschirm.
Im Gegensatz zur Funktion "K" hält der Cursor nicht in den mit "S" gekennzeichneten Feldern an, es sei denn, sie enthalten Fehlerkennzeichen.
In diesem Fall positioniert der Cursor auf das Fehlerkennzeichen.
Die Prüferin kann nun entweder

- die FELD-AUSL-Taste betätigen. Das Fehlerkennzeichen bleibt bestehen. Der Rest des Feldes wird nach einem weiteren Fehlerkennzeichen untersucht. Handelte es sich bereits um das letzte Zeichen des Feldes, werden die folgenden Felder nach einem Fehlerkennzeichen untersucht.

oder sie kann

- das Fehlerkennzeichen durch ein Tastaturzeichen ersetzen. Das eingegebene Zeichen wird angezeigt. Anschließend wird der Rest des Feldes nach weiteren Fehlerkennzeichen untersucht. Handelt es sich bei dem korrigierten Zeichen bereits um das letzte Zeichen des Feldes, werden die folgenden Felder nach einem Fehlerkennzeichen untersucht.

V Visuelle Prüfung mit Stop

Im PRÜF-Modus positioniert der Cursor auf die erste Stelle des mit "V" spezifizierten Prüffeldes. Es kann eine visuelle Prüfung erfolgen. Nach Betätigen der FELD-AUSL-Taste positioniert der Cursor zum nächsten Prüffeld bzw. nächsten Satz.

Wird im Parameter "PROFEN" keine Angabe gemacht, setzt das System automatisch die Funktion "S" ein.

	Eingabeformat

2.3.9 Korrektur-Prüfen (Parameter 8)

Wird Korrektur-Prüfen angegeben (Y = yes), so wird im PRÜF-Modus nach der Korrektur eines Zeichens (nach Betätigen der KORR-Taste) eine erneute Prüfung des gesamten Feldes gefordert.

2.3.10 Eingabezwang (Parameter 9)

Wird im Parameter "Eingabezwang" ein Y (yes) angegeben, besteht Eingabezwang. Das bedeutet, es muß mindestens ein Zeichen eingegeben werden, bevor die FELD-AUSL-Taste gedrückt wird. Wird die FELD-AUSL-Taste betätigt, ohne daß eine Dateneingabe erfolgte, wird ein Fehler signalisiert.

2.3.11 Beendigungszwang (Parameter 10)

Durch die Angabe Y (yes) wird festgelegt, daß ein Feld, in das mindestens ein Zeichen eingegeben wurde, bis zur vollen Länge mit Tastaturzeichen zu füllen ist. Soll keine Eingabe erfolgen, ist unmittelbar am Feldanfang die FELD-AUSL-Taste zu betätigen. Durch die Kombination der Parameter Eingabezwang und Beendigungszwang kann die Eingabe fester Feldlängen erzwungen werden (sinnvoll z.B. bei 4stelligen Postleitzahlen oder 8stelligen Bankleitzahlen).

2.3.12 Feldbegrenzung (Parameter 11)

Die Angabe von Y (yes) in diesem Parameter bewirkt, daß das Feld auf jeden Fall mit der FELD-AUSL-Taste abgeschlossen werden muß.

Die Anwendung der Funktion Feldbegrenzung ist dann sinnvoll, wenn verhindert werden soll, daß die Bedienungskraft versehentlich die Länge des Feldes überschreitet und die Dateneingabe im nächsten Feld fortsetzt. Ohne die Angabe Feldbegrenzung wird das Feld durch die Eingabe der max. möglichen Zeichenanzahl abgeschlossen.

Die Kombination von Eingabezwang, Beendigungszwang und Feldbegrenzung ist möglich.

	Eingabeformat

2.3.13 Rechtsbündig (Parameter 12)

Wird im Parameter "Rechtsbündig" ein Y (yes) angegeben, so werden die Daten im Eingabefeld rechtsbündig abgestellt.

Das bedeutet: Wird in einem Datenfeld eine Eingabe vorgenommen, befindet sich der Cursor auf der 1. Position des Feldes (links). Die Daten werden von links nach rechts entsprechend der Eingabe abgestellt.

Wird das Feld mit der FELD-AUSL-Taste verlassen, ohne daß die Dateneingabe bis zur vollen Feldlänge erfolgte, so werden die eingegebenen Daten nach rechts bis ans Ende des Eingabefeldes geschoben. Die restlichen unbenutzten Stellen (links) werden mit Füllzeichen aufgefüllt.

Die Anwendung dieser Funktion ist bei rein numerischen Daten sinnvoll.

Bemerkung:

Wird im Parameter "Rechtsbündig" keine Angabe gemacht, stellt das System die Daten automatisch linksbündig ab und füllt unausgenutzte Stellen des Datenfeldes (rechts) mit Füllzeichen auf.

2.3.14 Füllzeichen (Parameter 13/14)

Mit der Angabe des Füllzeichens wird festgelegt, welches Zeichen vom System in einem Eingabefeld abgestellt wird, das von der Bedienungskraft nicht bis zur vollen Länge durch Dateneingabe gefüllt wurde.

Die beiden möglichen Füllzeichen sind:

Z = Zero (Null)
S = Space (Blank)

Die Angabe "Z" bzw. "S" bezieht sich sowohl auf rechts- als auch auf linksbündig abgestellte Daten.

Das System unterscheidet zwischen Füllzeichen für Felder

- "Mit Daten" (ein oder mehrere Zeichen wurden eingegeben) und
- "Ohne Daten" (es wurde kein Zeichen eingegeben, sondern das Feld durch Betätigen der FELD-AUSL-Taste übergangen).

	Eingabeformat

Die Füllzeichen-Angabe "mit Daten" muß nicht identisch sein mit der Füllzeichen-Angabe "ohne Daten".

Beispiel: Mit Daten "Z"
Ohne Daten "S"

Wird keine Angabe über Füllzeichen gemacht, setzt die Anlage bei numerischen Feldern automatisch "Z" und sonst "S" ein.

2.3.15 Stapelsummen (Parameter 15/16)

Die Stapelsummenrechnung dient zur Kontrolle eingegebener numerischer Werte. Für diesen Zweck stehen bis zu 5 Akkumulatoren zur Verfügung. Zwei grundsätzlich unterschiedliche Methoden der Stapelsummenrechnung sind möglich:

- Durch ein gesondertes Eingabeformat wird zu Beginn der Erfassung die manuell ermittelte Stapelsumme eingegeben und in einen Akkumulator addiert (z.B. Gesamtstückzahl bei einer Auftragserfassung). Während der Erfassung werden die entsprechenden Felder vom Akkumulator subtrahiert.

- Es werden sowohl positive als auch negative Werte erfaßt und in den Akkumulator addiert (z.B. eine Saldenliste mit Soll- und Habensalden).

In beiden Fällen muß nach Abschluß des Stapels der Akkumulator den Wert Null haben, anderenfalls wird die Fehlermeldung "STAPELSUMMENDIFFERENZ" angezeigt.

Der Bediener hat die Möglichkeit, sich die Differenzen in der Erfassungsebene oder in der Supervisorebene anzeigen zu lassen.

Zusammenfassung:

- Parameter 15 gibt an, ob das Feld in den Akkumulator addiert (A), oder vom Akkumulator subtrahiert (S) werden soll.

- Parameter 16 gibt die Nummer des angesprochenen Akkumulators an.

- Pro Stapel können bis zu 5 verschiedene Stapelsummenrechnungen durchgeführt werden, da 5 14stellige Akkumulatoren zur Verfügung stehen. Die Fehlermeldung "STAPELSUMMENDIFFERENZ" erscheint, wenn einer oder mehrere der 5 Akkumulatoren ungleich Null ist. Um zu erkennen, in welchem Akkumulator der Fehler aufgetreten ist, müssen die Stapelsummendifferenzen vom Bediener angezeigt werden.

- Die Akkumulatoren können innerhalb eines Stapels beliebig oft von verschiedenen Formaten und Feldern angesprochen werden.

- Wird bei einer Stapelsummenrechnung eine Gesamtsumme vorgegeben, so muß hierfür ein eigenes Format erstellt werden, das bei der Erfassung nur einmal durchlaufen wird.

- Die Stapelsummenrechnung wird erst durchgeführt, wenn die Bedienungskraft die Funktion "BEENDEN" anwählt.

2.3.16 Tabelle (Parameter 17-19)

Das System unterscheidet zwischen Standardtabellen und Indextabellen.

2.3.16.1 Tabellenvergleich (Standardtabellen)

Beim Tabellenvergleich prüft das System Eingabefelder auf ihre Gültigkeit. Die Daten der Eingabefelder werden mit Werten, die in einer Tabelle auf der Magnetplatte gespeichert sind, verglichen.

Abhängig von den Erfordernissen der Anwendung kann eine Tabelle alle gültigen oder alle ungültigen Eingaben für ein Feld enthalten.

- Sind in einer Tabelle alle gültigen Eingaben für ein Feld abgespeichert, so erhält die Bedienungskraft bei dem Versuch Daten einzugeben, die nicht mit einem der Tabellenwerte übereinstimmen, eine Fehlermeldung, d.h. die Eingabe der Bedienungskraft ist ungültig.

- Enthält eine Tabelle jedoch alle ungültigen Eingaben für ein Feld, wird eine Fehlermeldung angezeigt, wenn die Eingabe mit einem der Tabellenwerte übereinstimmt.

	Eingabeformat

Im Parameter 17 des Eingabeformats wird die Nummer der Tabelle angegeben, die zum Vergleich mit dem entsprechenden Eingabefeld herangezogen wird. Parameter 18 gibt an, ob in der Tabelle alle gültigen oder ungültigen Werte gespeichert sind (geben Sie "Y" ein, wenn die Tabelle ungültige Werte enthält).

Um den Suchvorgang möglichst zu verkürzen, wird die Organisation der Tabelle (A, D oder R) im Parameter 19 angegeben.

A (ascending) - Die Werte der Tabelle sind in aufsteigender Reihenfolge gespeichert, z.B. 1, 2, 3, 4, 5.

D (descending) - Die Werte der Tabelle sind in absteigender Reihenfolge gespeichert, z.B. 5, 4, 3, 2, 1.

R (random) - Die Werte der Tabelle sind in unsortierter Folge gespeichert, z.B. 5, 3, 1, 2, 4.

Wird keine Angabe über die Organisation der Tabelle gemacht, wird vom System unsortierte Folge angenommen.

Während des Tabellenvergleichs befindet sich der Cursor hinter der letzten Stelle des Vergleichsfeldes.

- Wird ein Tabellen-Vergleichsfehler festgestellt, positioniert der Cursor erneut auf die erste Stelle der Vergleichsfeldes. Kann der richtige Wert nicht sofort ermittelt werden, kann die Bedienungskraft das Feld duch Setzen eines Fehlerkennzeichens übergehen und normal weiterarbeiten.

- Wird kein Vergleichsfehler festgestellt, positioniert der Cursor auf die erste Stelle des nächsten Feldes.

Bemerkung:

Das Anlegen einer Tabelle wird im Supervisor-Handbuch in dem Kapitel "Tabellen-Zuweisung" beschrieben.

	Eingabeformat

Untertabellen

Bis zu 20 Tabellen können zur gleichen Zeit zugewiesen werden. Besteht Bedarf nach zusätzlichen Tabellen, können innerhalb einer Tabelle zwei oder mehrere Untertabellen angelegt werden. Voraussetzung ist, daß alle Untertabellen unterschiedliche Feldlängen aufweisen. Da jede Tabelle aus einem Datenstapel erzeugt wird, müssen die Untertabellen innerhalb dieses Stapels durch unterschiedliche Eingabeformate oder unterschiedliche Felder innerhalb eines eingabeformates repräsentiert werden.

Beispiel:

746AE52	1957A	Format 1
728DC52	1958A	Format 1
346BA52	1961A	Format 1
Feld 1	Feld 2	

Aufbau der Tabelle mit unterschiedlichen Feldern innerhalb eines Eingabeformates

746AE52	Format 1
728DC52	Format 1
346BA52	Format 1
1957A	Format 2
1958A	Format 2
1961A	Format 2
Feld 1	

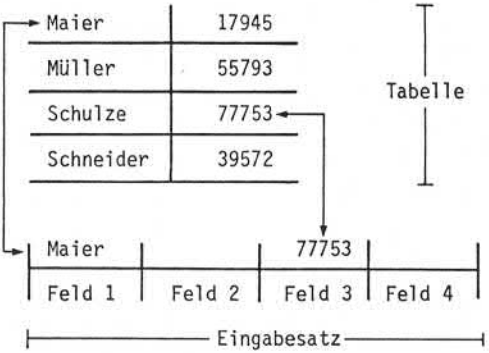
Aufbau der Tabelle mit einem Feld und zwei Eingabeformaten.

	Eingabeformat

Maßgebend für die Auswahl der Tabellenwerte ist in beiden Fällen die Länge des Vergleichsfeldes.

Wird z.B. ein Vergleich mit einem 7stelligen Feld gestartet, berücksichtigt das System nur alle 7stelligen Tabellenwerte. Bei einem Vergleich mit einem 5stelligen Feld werden dementsprechend nur die 5stelligen Tabellenwerte abgeprüft.

Beispiel:



2.3.16.2 Indextabellen

Standardmäßig sind im System die Tabellen 16-20 als Indextabellen zugeordnet.

Während der Generierung des Betriebssystems kann diese Anzahl bis auf 20 Indextabellen erweitert werden. Die Nummern der Indextabellen werden absteigend vergeben. Bei der Zuordnung von 5 zusätzlichen Indextabellen erhalten diese die Nummern 11-15.

Die Verwendung von Indextabellen bietet zwei wesentliche Vorteile:

- Durch den Index wird der Zugriff auf sehr große Tabellen wesentlich beschleunigt.

	Eingabeformat

- Werte aus der Indextabelle können vom Eingabeprogramm auf dem Bildschirm angezeigt und/oder in den Eingabesatz übernommen werden. Das wird ermöglicht durch die Zuordnung eines max. 99stelligen Korrespondenzfeldes zum Schlüsselbegriff. Schlüsselbegriff und Korrespondenzfeld müssen unterschiedliche Längen haben, da anderenfalls das Korrespondenzfeld als Schlüssel interpretiert wird. Sind in der Indextabelle nur Schlüsselfelder gespeichert, erfolgt die Verarbeitung wie bei den Standard-Tabellen. Die Verwendung von Untertabellen ist nicht erlaubt.

Beispiel für den Aufbau einer Indextabelle mit Korrespondenzfeldern:

588230	000950
588428	004500
652300	012500
Art.-Nr.	Preis

Falscher Aufbau der Tabelle. Das Feld Preis wird als Schlüsselfeld interpretiert, da es die gleiche Länge wie das Feld Artikelnummer hat.

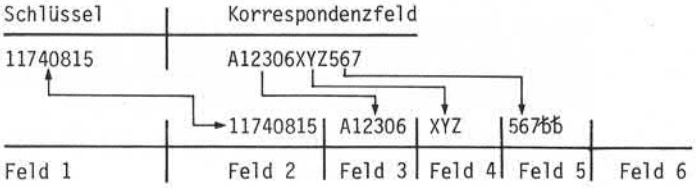
588230	0000950
588428	0004500
652300	0012500
Art.-Nr.	Preis

Richtiger Tabellenaufbau. Das Korrespondenzfeld hat eine andere Länge als der Schlüsselbegriff Artikelnummer.

Eingabeformat

Wird das Korrespondenzfeld in den Eingabesatz übernommen, beginnt die Übernahme mit dem auf das Vergleichsfeld folgenden Feld und endet, sobald alle Zeichen des Korrespondenzfeldes übernommen sind. Ist kein Korrespondenzfeld vorhanden, erfolgt eine Fehlermeldung. Ist das Korrespondenzfeld länger als der Rest des Eingabesatzes, wird ebenfalls ein Fehler angezeigt. Ein Feld des Eingabesatzes, das nicht vollständig belegt wird, wird rechts mit Leerzeichen aufgefüllt.

Beispiel:



Eingabe der Parameter für eine Indextabelle:

- Tabellennummer (Parameter 17) — 1-20, abhängig von der Systemgenerierung
- Ungültige Werte (Parameter 18) — muß freibleiben
- Organisation (Parameter 19) — A, D oder R
  Dieser Parameter hat eine andere Bedeutung als beim Tabellenvergleich.

  A = Vergleich mit dem Schlüsselbegriff. Eventuell vorhandene Korrespondenzfelder werden ignoriert.

  D = Vergleich mit dem Schlüsselbegriff und Anzeige des Korrespondenzfeldes ab der Fehlerzeile. Ist kein Korrespondenzfeld vorhanden, erfolgt eine Fehlermeldung.

  R = Vergleich mit dem Schlüsselbegriff und Übernahme des Korrespondenzfeldes in den Eingabesatz mit gleichzeitiger Anzeige.

	Eingabeformat

Folgende Punkte müssen bei der Anwendung von Indextabellen beachtet werden:

- Die Tabelle muß aufsteigend sortiert sein.
- Die Tabelle darf keine gelöschten Sätze enthalten.
- Es sollten keine doppelten Schlüsselbegriffe vorhanden sein.
- Schlüssel und Korrespondenzfeld müssen ≠ lang sein.
- Untertabellen sind nicht möglich.

2.3.17 Grenzprüfung (Parameter 20)

Mit der Grenzprüfung kann man für ein ganz bestimmtes Feld festlegen, daß der Eingabewert eine obere und eine untere Grenze nicht überschreiten darf. Grenzprüfungen werden sinnvollerweise nur bei rein numerischen Feldern durchgeführt.

Soll mit Grenzprüfung gearbeitet werden, hat man die Wahl zwischen den Angaben "I" und "O".

I bedeutet: Der Eingabewert muß innerhalb der Minimal- und Maximal-Werte liegen.

O bedeutet: Der Eingabewert muß außerhalb der Minimal- und Maximal-Werte liegen.

Die Minimal- und Maximal-Werte werden unter Parameter 21 und 22 eingetragen. Ihre Länge ist auf 14 Stellen begrenzt.

Bemerkung:

Die Grenzprüfung wird nur durchgeführt bei Eingabe von Daten. Bei Feldinhalt = Füllzeichen ohne Daten (Parameter 14) wird diese Prüfung nicht durchgeführt.

2.3.18 Auto-Funktion (Parameter 23)

Folgende Funktionen können vom System während der Datenerfassung automatisch ausgeführt werden:

- S Überspringen des Feldes (skip)

  Das Feld wird übersprungen und entsprechend der Füllzeichenangabe (Parameter 14) mit Leerzeichen (Blanks) oder Nullen gefüllt.

Eingabeformat

- D Duplizieren (duplicate)

  Die Daten des korrespondierenden Feldes (gleiche Feldnummer) des vorhergehenden Satzes werden in dieses Feld übernommen.

- I Automatisches Zählen bzw. Erhöhen (increment)
  Der Inhalt des korrespondierenden Feldes (gleiche Feldnummer) des vorhergehenden Satzes wird um 1 erhöht und in dieses Feld übernommen.

- E Übernehmen der Maske in den Eingabesatz (emit)
  Die Zeichen der Maske (Feldbeschreibung) werden in dieses Feld übernommen. Die Feldlänge muß entsprechend der Maske definiert werden.

Alle Funktionen werden in Abhängigkeit der AUTO-Taste durchgeführt:

- Ist die AUTO-Taste gesetzt (AUTO wird in der Statuszeile angezeigt), wird die Funktion ausgeführt.

- Ist die AUTO-Taste nicht gesetzt, erscheint zwar in der Statuszeile ein Hinweis auf den Typ der Funktion (SPRG, DUP, ERH, EINF), die für das anstehende Feld ausgeführt werden sollte, aber die Funktion wird nicht ausgeführt. Stattdessen hat man die Möglichkeit, in dieses Feld Daten einzugeben.

  Bemerkung:

  Bei den Funktionen "Duplizieren" und "Automatisches Zählen" greift das System auf den Feldinhalt zurück, der im vorherigen Satz unter der gleichen Feldnummer gespeichert ist. FMT-Nummern bleiben dabei unberücksichtigt.
  Vorsicht!

	Eingabeformat

2.3.19 Prüfziffer (Parameter 24-26)

Eine Prüfziffer gewährleistet mit hoher Wahrscheinlichkeit, daß die Daten eines Feldes korrekt eingegeben werden. Die Prüfziffer ist die letzte Stelle eines numerischen oder alphanumerischen Feldes und wird aus dem Rest des Feldes durch ein spezielles Rechenverfahren ermittelt.

Folgende Eingaben sind in den Parametern 24-26 des Eingabeformates erforderlich, falls eine Prüfziffernrechnung gewünscht wird:

- Nummer (Parameter 24)

  Da im System bis zu 15 verschiedene Verfahren zur Errechnung der Prüfziffer gespeichert werden können, muß in diesem Parameter die Nummer des gewünschten Verfahrens angegeben werden.

- Methode (Parameter 25)

  Zwei grundsätzlich unterschiedliche Funktionen der Prüfziffernrechnung sind möglich:

  G bedeutet generieren (erzeugen). Das System errechnet für die letzte Stelle eines Datenfeldes eine Prüfziffer, egal ob auf dieser Stelle bereits ein Zeichen eingegeben wurde oder nicht.
  Das Generieren von Prüfziffern für Datenfelder ist ein Vorgang, der normalerweise zeitlich vor der Datenerfassung liegt. Er dient dazu, Prüfziffern (die später ein Bestandteil des Datenfeldes sind) zunächst einmal zu errechnen.

  V bedeutet verifizieren (prüfen). Der Eingabewert wird zusammen mit der Prüfziffer eingegeben. (Die Prüfziffer ist die letzte Stelle des Wertes).
  Das System errechnet für das entsprechende Feld die Prüfziffer. Ist die errechnete Prüfziffer identisch mit der eingetasteten, positioniert der Cursor in das nächste Feld. Bei Ungleichheit meldet das System PRUEFZIFFERN-FEHLER.

	Eingabeformat

- Mehrere Felder (Parameter 26)

  Eine Prüfziffernrechnung kann über mehrere Felder durchgeführt werden. Das System reiht die Felder intern aneinander und führt für den gesamten Wert eine Prüfziffernrechnung durch.

  Bei allen Feldern, die zur Prüfziffernrechnung über mehrere Felder herangezogen werden, wird im Eingabeformat unter Punkt 26 ein "Y" eingesetzt. Nur das letzte Feld der Reihe gibt die Nummer der Prüfziffernrechnung und die Methode an.

  Zwischen Feldern, die für die Prüfziffernrechnung gesammelt werden, darf kein Feld sein, in dem die Frage "MEHRERE FELDER?" mit Nein beantwortet wurde.

  Bei einer Prüfziffernrechnung über mehrere Felder ist es unbedingt erforderlich, daß im letzten Feld, das zur Prüfziffernrechnung herangezogen wird, Daten eingegeben werden. Zweckmäßigerweise sollte für dieses Feld EINGABEZWANG spezifiziert werden.

- Eine genaue Erläuterung der Prüfziffernverfahren finden Sie in Kapitel 8.

2.3.20 Update (Parameter 27)

Die Anwendung der UPDATE-Funktion ist sinnvoll, wenn bestimmte Felder eines Satzes aktualisiert werden müssen, während die restlichen Daten unverändert bleiben sollen.

Die Funktion wird im UPDATE-Modus wirksam, d.h. nach Anwahl der Funktion UPDATE aus der Daten-Erfassungs-Übersicht. Der Cursor positioniert unmittelbar auf die einzelnen UPDATE-Felder, in denen die aktuellen Daten eingegeben werden können. Alle anderen Felder bleiben unverändert.

	Eingabeformat

Folgende Spezifikationen sind für ein UPDATE-Feld möglich:

C Tritt bei Durchführung einer Stapelkontrollrechnung mit diesem Feld eine Stapelsummendifferenz auf, müssen die mit "C" spezifizierten Update-Felder durch Neueingabe geprüft werden.
(Diese Funktion arbeitet analog der Angabe "C" Abhängigkeitsprüfung im Prüf-Modus).

E Im UPDATE-Modus kann in diesem Feld eine Eingabe erfolgen. (Der ursprüngliche Wert dieses Feldes wird nicht mehr angezeigt).

K Im UPDATE-Modus wird der Wert eingegeben und mit dem ursprünglichen Wert verglichen. (Diese Funktion entspricht der Funktion K im PRÜF-Modus).

S Der Inhalt des Update-Feldes wird zur Sichtprüfung angezeigt und kann nicht verändert werden.

Änderungen sind in den mit "S" gekennzeichneten Update-Feldern nur an den Stellen möglich, die ein Fehlerkennzeichen enthalten. Die Fehlerkennzeichen können durch ein Tastaturzeichen ersetzt, oder mit der FELD-AUSL-Taste übergangen werden.
(Diese Funktion entspricht der Funktion "S" im Prüf-Modus).

V Der Inhalt des Feldes wird angezeigt. Eingaben sind nur nach Betätigung der KORR-Taste möglich.

R Diese Angabe beendet die UPDATE-Funktion für diesen Satz. Das System verzweigt zum 1. UPDATE-Feld des nächsten Satzes.

Wird keine der möglichen Funktionen angegeben, setzt das System automatisch die Funktion "S" ein.

2.3.21 Aufsteigende Folge (Parameter 28)

Bei Angabe von Y (yes) prüft das System, ob der Inhalt des Eingabefeldes größer oder gleich dem Inhalt des korrespondierenden Feldes (gleiche Feldnummer) im vorhergehenden Satz ist. Ist der Inhalt kleiner, wird ein Fehler angezeigt.
Diese Funktion kann auch bei alphanumerischen Feldern angewandt werden.

Bemerkung:

Das System greift auf den vorhergehenden Satz zu. FMT-Nummern bleiben dabei unberücksichtigt.
Vorsicht!

Eingabeformat

2.3.22 Tabulationsfeld (Parameter 29)

Bei der Datenerfassung ist es vorteilhaft, wenn weniger benutzte Felder oder Feldgruppen auf Wunsch übersprungen werden können.

Zu diesem Zweck wird für Felder, in die im Regelfall Eingaben erfolgen, der Parameter 29 mit Y (yes) angegeben. Die Bedienungskraft kann durch Betätigen der TAB-Taste oder der Tastenkombination SATZ-T von jedem beliebigen Feld des Eingabesatzes zum nächsten Tabulationsfeld springen.

Voraussetzung ist, daß für kein dazwischenliegendes Feld Eingabezwang angegeben ist. In diesem Fall stoppt das System mit der Meldung "EINGABEZWANG". Beim Tabulieren übersprungene Felder werden mit den im Parameter 14 angegebenen Füllzeichen gefüllt. In einem Eingabesatz können beliebig viele Tabulationsfelder definiert werden.

2.3.23 Zusatzfeld (Parameter 30)

Diese Funktion erlaubt, die Sätze eines bestehenden Stapels um eine beliebige Anzahl von Feldern zu erweitern.

Voraussetzung ist, daß der Stapel auf Magnetband ausgegeben und anschließend mit dem erweiterten Eingabeformat auf die Platte zurückgelesen wird.
Das folgende Beispiel zeigt den Ablauf:

- Ein Satz des Datenstapels

  Artikelnummer (6)
  Bezeichnung (20)
  Preis (5)
  Bestand (8)

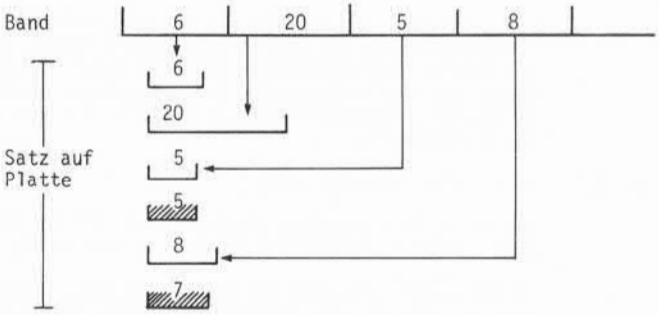
- Jeder Satz des Datenstapels soll nach dem Preis um ein 5stelliges Feld und nach dem Bestand um ein 7stelliges Feld erweitert werden.

- Der Stapel wird auf Band ausgegeben:

	Eingabeformat

- Das Eingabeformat wird geändert, und zwar wird ein 5stelliges und 7stelliges Feld eingefügt, unter Angabe der für diese Felder gültigen Feldbedingungen und Masken. Beide Felder erhalten im Parameter 30 (Zusatzfeld) die Angabe Y (yes). Außerdem ist es sinnvoll, die Zusatzfelder als UPDATE-Felder zu kennzeichnen, weil dadurch die nachträgliche Erfassung erleichtert wird.

- Die Daten werden vom Band mit dem neuen Eingabeformat auf die Platte zurückgeschrieben.



Während der Übertragung Band - Platte entstehen Lücken im Datensatz, die mit den im Parameter 14 angegebenen Zeichen gefüllt werden.

- Die Bedienungskraft kann nun in den Zusatzfeldern die notwendigen Eingaben vornehmen.

2.3.24 Kettung (Parameter 31)

Soll ein Stapel mit mehreren Eingabeformaten erfaßt werden, muß zu Beginn der Erfassung oder im Standard-Job die Verkettung (Aufeinanderfolge) der einzelnen Eingabeformate festgelegt werden.

Durch Angabe von Y (yes) im Parameter 31 (Kettung) wird der Bedienungskraft erlaubt, von der festgelegten Verkettung abzuweichen.
Die letzte Stelle des zum Kettfeld erklärten Feldes gibt die Nummer des Eingabeformates an, mit dem der nächste Satz erfaßt werden soll.

Es kann auch ein separates 1stelliges Kettfeld spezifiziert werden.

	Eingabeformat

Die Angabe "Kettung" muß nicht die letzte Angabe eines Satzes sein, jedoch wird die Kettfunktion erst wirksam, wenn der Eingabesatz vollständig eingegeben ist.

Gibt die Bedienungskraft in einem Kettfeld anstelle einer Zahl ein ß (Blank) ein, wird die bei Stapelbeginn festgelegte Formatreihenfolge eingehalten.

Wird ein Stapel forgesetzt, bei dem im letzten Satz eine Kettangabe gemacht wurde, wird diese Angabe nicht berücksichtigt. Es wird immer das Format benutzt, das laut festgelegter Verknüpfung als nächstes benutzt werden soll.

Bemerkung:

Durch den LINK-Befehl ist eine programmabhängige Anwahl des nächsten Eingabeformates möglich.

2.3.25 Feldende-Editor (Parameter 32)

Für jedes Feld des Eingabesatzes, dessen Parameter 32 mit Y (yes) angegeben ist, wird das im Standard-Job angegebene Feldende-Editorprogramm durchlaufen.

Bevor das System den Feldende-Editor für ein Eingabefeld durchläuft,

- werden alle nachfolgenden Felder des Datensatzes mit den im Eingabeformat spezifizierten Füllzeichen gefüllt. Somit ist im Feldende-Editor der gesamte Satz ansprechbar. Es können auch die Felder bearbeitet werden, in die noch keine Eingabe erfolgt ist.

- werden alle Prüfungen des Eingabeformats in folgender Reihenfolge durchgeführt:

  - Aufsteigende Folge
  - Prüfziffern
  - Grenzprüfung
  - Tabellenvergleich

Eine genaue Erläuterung des Feldende-Editors finden Sie im Abschnitt 3.1.

2.3.26 Feld einfügen/löschen (Parameter 33)

Die Angabe von Y (yes) in diesem Parameter erlaubt, durch Betätigen der Taste EINF bzw. LÖ in Verbindung mit der FELD-Taste, über Feldgrenzen hinaus Zeichen einzufügen oder zu löschen.

Ein bei Betätigung der EINF-Taste rechts aus dem Feld herausgeschobene Zeichen geht nicht verloren, sondern wird auf die erste Stelle des folgenden Feldes transportiert. Dieses Feld wird ebenfalls um eine Stelle nach rechts verschoben und das überlaufende Zeichen auf die erste Stelle des darauf folgenden Feldes gesetzt. Das zeichenweise Verschieben endet beim ersten Feld, dessen Parameter 33 nicht mit Y gekennzeichnet ist. Eine beliebige Anzahl von Feldern kann mit dieser Funktion versehen werden.

Beispiel:

A B C D E F G H I J	Parameter 33 = Y	Satz vor dem Einfügen eines Zeichens
7 4 1 2 3	Parameter 33 = Y	
X Y Z	Parameter 33 = Y	
2 3 5 1 6	Parameter 33 ≠ Y	
A B C X D E F G H I	Parameter 33 = Y	Satz nach Einfügen des Zeichens "X" in Feld 1. Das letzte Zeichen aus Feld 3 geht verloren.
J 7 4 1 2	Parameter 33 = Y	
3 X Y	Parameter 33 = Y	
2 3 5 1 6	Parameter 33 ≠ Y	

Analog zum Einfügen eines Zeichens wird beim Löschen in die freiwerdende letzte Stelle des Feldes das erste Zeichen des folgenden Feldes transportiert. Dieser Vorgang setzt sich bis zum ersten Feld fort, dessen Parameter 33 nicht mit Y gekennzeichnet ist.

	Eingabeformat

Beispiel:

A B C D E F G H I J	Parameter 33 = Y	Satz vor dem Löschen eines Zeichens
7 4 1 2 3	Parameter 33 = Y	
X Y Z	Parameter 33 = Y	
2 3 5 1 6	Parameter 33 ≠ Y	
A B C E F G H I J 7	Parameter 33 = Y	Satz nach Löschen des Zeichens "D" im Feld 1. Die letzte Stelle in Feld 3 wird mit einem Leerzeichen gefüllt.
4 1 2 3 X	Parameter 33 = Y	
YZb	Parameter 33 = Y	
2 3 5 1 6	Parameter 33 ≠ Y	

Hinweis:

Die Operation Feld einf/löschen ist bei der Eingabe von Programmen erlaubt. Das bedeutet, daß innerhalb eines Satzes Programmzeilen eingefügt oder gelöscht werden können.

2.4 Beispiel eines Eingabeformates

Auf den folgenden Seiten finden Sie ein Beispiel für die Eingabeformate der Auftragserfassung, die im Abschnitt 2.2 vorgestellt wurde.

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name AUF1 Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn Zeile	Masken-Beginn Spalte	Maske mit ≠ beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Kon.-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Fällzeich: Mit Daten	Fällzeich: Ohne Daten	Stap. Sum.: Add./Sub.	Stap. Sum.: Nummer	Tabelle: Nummer	Tabelle: Ung. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-Funktion	Prül-Ziffer: Nummer	Prül-Ziffer: Methode	Prül-Ziffer: Mehr. Felder	Update	Aufsteig. Folgs	Tab.-Feld	Zusatz-Feld	Kettung	Feldende-Echo	Feld einf./löschen
1	1	5	AUFTRAG NR.: ≠	6	N	K	Y	Y	Y																				
2	1	26	DATUM: ≠	2	N	R					Y								I											
3	1	35	.≠	2	N						Y								I										
4	1	38	.≠	2	N					Y																			
5	2	1	NAME: ≠	20	L			Y			Y																		
6	3	1	STRASSE: ≠	20	L			Y	Y		Y																		
7	4	1	ORT: ≠	20	L			Y	Y		Y																		
8	6	1	TELEFON: ≠	12	L						Y																		
9	6	26	KD.-NR.: ≠	6	N						Y																		

Feld-Nr.	21 Min-Wert	22 Max-Wert
2	1	31
3	1	12

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name AUFZ Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn *		Maske mit ≠ beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Kon.-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füll-zeich	Stap. Sum.	Tabelle			Grenzprüfg	Auto-Funktion	Prüf-Ziffer			Update	Auftrag. Folge	Tab.-Feld	Zusatz-Feld	Kettung	Feldende-Editor	Feld einl. /löschen
	Zeile	Spalte										Mit Daten	Ohne Daten	Add./Sub.	Nummer	Nummer	Ung. Wert	Organis.			Nummer	Methode	Mehr. Felder						
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	23	24	25	26	27	28	29	30	31	32	33
0	1	1	ARTIKELNUMMER MENGE																										
1	2	4	≠	7	N	K	Y	Y	Y							1		A											
2	2	17	≠	5	N	C								A	1														
3	4	1	BEI AUFTRAGSENDE "3" EINGEBEN: ≠	1	N								S														Y		

Feld-Nr.		21	Min-Wert		22	Max-Wert

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name AUF3 Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit ≠ beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Kon. Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Fill-zeich	Mit Daten	Ohne Daten	Stap. Sum.	Add./Sub.	Tabelle: Nummer	Tabelle: Ung. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-Funktion	Prüf-Ziffer: Nummer	Prüf-Ziffer: Methode	Prüf-Ziffer: Mehr. Felder	Update	Auftrag. Folge	Tab. Feld	Zusatz-Feld	Kettung	Feldende-Editor	Feld eint. löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	13	14	15	15	16	17	18	19	20	23	24	25	26	27	28	29	30	31	32	33
1	1	1	GESAMT STUECK: ≠	6	N	C		Y			Y				S	1																

Feld-Nr.		21	Min-Wert		22	Max-Wert

Programmstruktur

3. Programmstruktur

Zusätzlich zu den durch das Eingabeformat oder durch Dienstprogramme ausgeführten Funktionen hat der Anwender des Systems 620/45 die Möglichkeit, individuelle Programmroutinen zu erstellen.

Anwenderprogramme gliedern sich - abhängig von ihrer Funktion - in 4 unterschiedliche Gruppen:

- Feldende-Editor
- Stapelende-Editor
- Ausgabeprogramm
- Sortierprogramm

Alle Anwenderprogramme sind unter einem max. 8stelligen Namen in Programmbibliotheken gespeichert. Der Name kann aus beliebigen Tastaturzeichen bestehen, nur die erste Stelle muß ein Buchstabe sein.

Einige Programmbefehle dürfen nur in bestimmten Programm-Typen benutzt werden. Die Abhängigkeit zeigt folgende Tabelle:

	Feld	Stapel	Ausg.	Sort
BYPASS	N	J	J	J
POSITION	J	N	N	N
SET	J	N	N	N
LOAD	J	N	N	N
WHEN FIELD	J	N	N	N
OUTPUT *)	N	J	J	N
SORT	N	N	N	J

*) Im Stapelende-Editor erzeugt der OUTPUT-Befehl eine Fehlerliste auf der Magnetplatte.

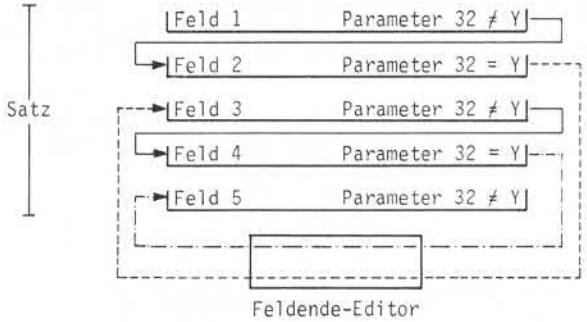
	Programmstruktur

3.1 Feldende-Editor

Der Feldende-Editor erweitert die vom Eingabeformat vorgenommene Feldprüfung und Bearbeitung. Das Programm erlaubt die Abhängigkeitsprüfung von Feldern, die Übernahme von Feldinhalten aus Indexdateien, die Aktualisierung von Indexdateien parallel zur Erfassung und die Ausgabe von Daten auf dem Arbeitsplatzdrucker.

Der Feldende-Editor wird nach beendeter Eingabe von Feldern durchlaufen, deren Parameter 32 im Eingabeformat mit Y (yes) angegeben ist.

Beispiel:



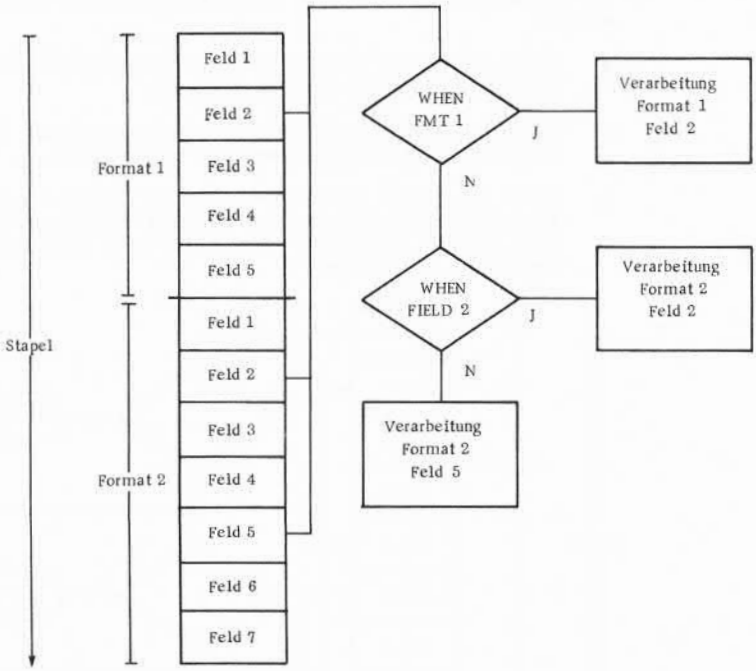
Da für einen Erfassungslauf nur ein Feldende-Editor angegeben werden kann, wird von allen Feldern, deren Parameter 32 mit Y (yes) angegeben ist, dieser Feldende-Editor angesprungen. Das gilt auch bei mehreren Eingabeformaten innerhalb des Stapels.

Daher ist es erforderlich, im Feldende-Editor eine Unterscheidung nach Eingabeformat und Feldnummer zu treffen.

Programmstruktur

Durch die Bedingungsabfrage WHEN FMT n kann zwischen Eingabeformaten, durch die Abfrage WHEN FIELD n zwischen Feldern innerhalb eines Eingabeformats unterschieden werden.

Beispiel:



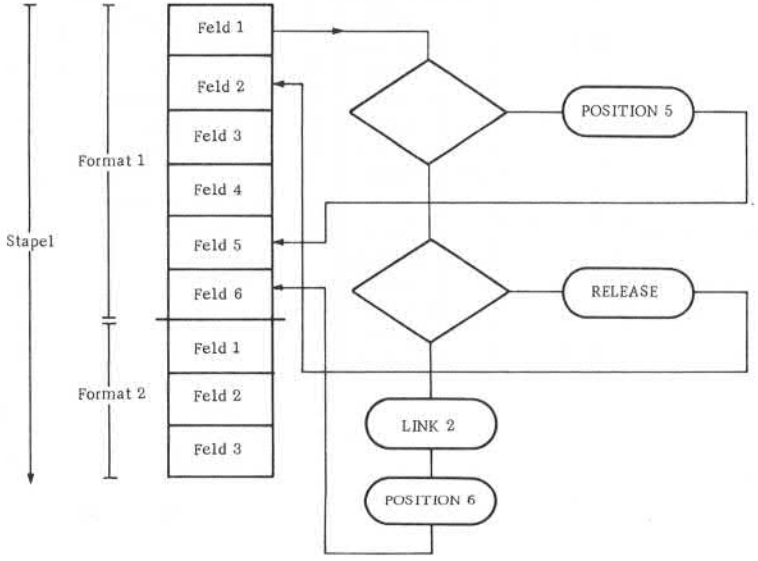
	Programmstruktur

Während der Ausführung des Feldende-Editors stehen alle Felder des Eingabesatzes im Zugriff. Noch nicht eingegebene Felder sind mit den im Parameter 14 des Eingabeformats angegebenen Zeichen gefüllt. Dadurch sind Querprüfungen zwischen Feldern und die Veränderung von Feldinhalten möglich.

Zusätzlich kann durch Verwendung der BACK-Instruktion auf den Inhalt vorangegangener, bereits erfaßter Sätze zugegriffen werden (siehe Punkt 5.6.4.1).

Der Feldende-Editor kann auf 3 verschiedene Arten verlassen werden:

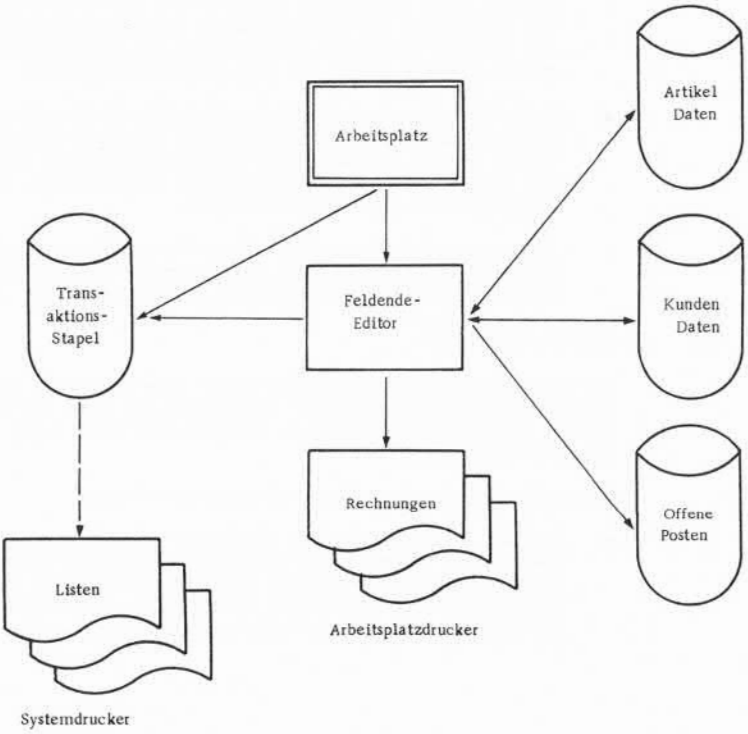
- Durch den Befehl RELEASE wird die Erfassung mit dem unmittelbar folgenden Feld fortgesetzt.
- Durch den Befehl POSITION wird die Erfassung auf dem im POSITION-Befehl angegebenen Feld fortgesetzt.
- Durch den Befehl LINK wird auf das im Befehl angegebene Eingabeformat verzweigt. Auf den LINK-Befehl muß POSITION oder RELEASE folgen. Der LINK-Befehl wird erst am Ende des Satzes ausgeführt.



	Programmstruktur

Der Feldende-Editor bietet neben der Prüfung und Manupulation von Eingabedaten auch die Möglichkeit der Echzeit-Verarbeitung (Real-time-mode). Das bedeutet, der Feldende-Editor aktualisiert Stammdateien anhand der eingegebenen Daten, oder produziert Ergebnisse aus Eingabedaten und Stammdaten. Diese Prozesse laufen im Gegensatz zur Stapelverarbeitung nicht nach, sondern während der Datenerfassung ab. Die Echtzeit-Verarbeitung wird durch Anwahl der Funktion DATEI-BEARBEITUNG gestartet.

Beispiel für eine Echtzeit-Fakturierung:



Programmstruktur

Ablauf:

- Zum Tagesanfang wird das Fakturierprogramm durch Anwahl der Funktion DATEI-BEARBEITUNG und des entsprechenden Standard-Jobs gestartet.
- Erscheint ein Kunde, wird nach Eingabe der Kundennummer der zugehörige Kundenstammsatz gelesen und der Rechnungskopf gedruckt.
- Die Bedienungskraft gibt Nummer und Menge der gewünschten Artikel ein. Das System prüft anhand der Artikelstammdatei die Gültigkeit der Artikelnummer und den Lagerbestand. Ist der Artikel vorhanden, wird sofort die Rechnungszeile gedruckt und die Menge vom Lagerbestand abgebucht.
- Nach Fakturierung aller Artikelpositionen druckt das System automatisch den Rechnungsabschluß. Ein ggf. einzuräumender Rabatt ist im Kundenstammsatz gespeichert. Der Jahresumsatz des Kunden wird im Kundenstammsatz aktualisiert. Die Offene-Posten-Datei wird fortgeschrieben.
- Die im Transaktionsstapel gespeicherten Bewegungsdaten können entweder an ein zentrales Rechenzentrum übertragen, oder dezentral mit Stapelprogrammen weiterverarbeitet werden.

3.2 Stapelende-Editor

Im Gegensatz zum Feldende-Editor, der eine sofortige Prüfung und Verarbeitung der eingegebenen Daten erlaubt, wird der Stapelende-Editor erst beim Beenden eines Stapels oder nach Aufruf durch den Benutzer durchlaufen.

Die Anwendung des Stapelende-Editors dient dazu,

- Fehler, die vom Programm im Stapel erkannt werden und nicht unmittelbar während der Erfassung korrigiert werden sollen, nach vollständiger Eingabe des gesamten Stapels in einer Fehlerliste auf der Platte aufzulisten, mit Fehlerkennzeichen zu versehen oder durch Fehlermeldungen anzuzeigen.

Programmstruktur

- Satz-zu-Satz-Prüfungen durchzuführen.
- Bestimmte Feldinhalte bis zum Ende des Stapels zu kumulieren und auf diese Weise eine Stapelsummenbildung zu ermöglichen, bei der z.B. die Differenz auf dem Bildschirm angezeigt werden kann.

Typische Prüfungen, die im Stapelende-Editor durchgeführt werden, sind:

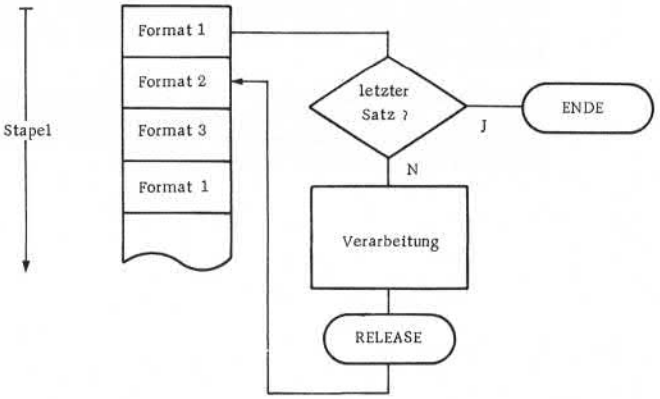
- umfangreiche Querkontrollen und -vergleiche über mehrere Sätze
- umfangreiche Grenzprüfungen
- Satz-zu-Satz-Prüfungen
- Zwischen- und Stapelsummenbildung
- Inhaltsprüfungen

Neben den erwähnten Prüf-Funktionen kann der Stapelende-Editor auch zur Aktualisierung von Stammdateien dienen, bei denen eine Echtzeitverarbeitung entweder nicht erforderlich oder nicht erwünscht ist.

Der Stapelende-Editor wird in der Datenerfassungsebene nur nach Aufruf der Funktion BEENDEN durchlaufen. Beim UNTERBRECHEN eines Stapels wird der Stapelende-Editor nicht angesprochen.

Der Stapelende-Editor beginnt bei der Abarbeitung mit dem 1. Satz eines Stapels; durch den RELEASE-Befehl wird jeweils der nächste Satz verarbeitet.

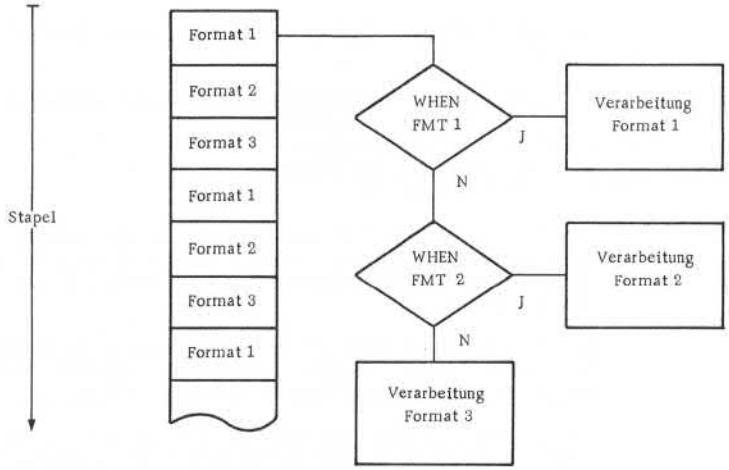
Beispiel:



	Programmstruktur

Wurden die Sätze eines Stapels mit unterschiedlichen Eingabeformaten erfaßt, muß im Stapelende-Editor mit Hilfe des Befehls WHEN FMT zwischen den verschiedenen Formaten unterschieden werden.

Beispiel:



Im Stapelende-Editor ist neben der Verarbeitung eines Stapels auch die Verarbeitung mehrerer Stapel möglich (Multi-Batch). Durch die Abfrage WHEN FILE erkennt das Programm den Beginn eines neuen Stapels. Soll die Verarbeitung eines Stapels abgebrochen und zum nächsten Stapel übergangen werden, ist der Befehl BYPASS zu codieren.

Multi-Batch-Verarbeitung ist nur möglich, wenn der Stapelende-Editor als unabhängiges Programm in der Supervisor-Ebene aufgerufen wird. Bei der Angabe des Stapelnamens muß mit Sternchen-Vereinbarung gearbeitet werden.

Programmstruktur

Werden vom Stapelende-Editor Fehler im Stapel erkannt, können diese auf 3 Arten signalisiert werden:

- Durch den Ausgabebefehl OUTPUT wird eine Fehlerliste erstellt. Diese Fehlerliste enthält alle im Stapel aufgetretenen Fehler. Nach dem Durchlauf des Stapelende-Editors wird die Liste am Bildschirm angezeigt, kann vom Supervisor aber auch auf Band oder Drucker ausgegeben werden.

  Während der Bearbeitung bzw. Korrektur der Fehler kann die Bedienungskraft die Fehlerliste beliebig oft aufrufen und sich am Bildschirm anzeigen lassen.
  Wird nach Korrektur der Fehler der Stapel beendet, erfolgt ein erneuter Durchlauf des Stapelende-Editors, und wird - sofern sich im Stapel noch Fehler befinden - automatisch eine neue Fehlerliste erstellt.

  Eine Fehlerliste kann nicht separat gelöscht werden, da sie unmittelbar zum Stapel gehört.

- PAUSE/SHOW (nur Anzeige)

- Setzen Fehlerkennzeichen mit FLAG-Befehl

3.3 Abgrenzung zwischen Feldende- und Stapelende-Editor

Die Unterschiede zwischen Feldende- und Stapelende-Editor bestehen im wesentlichen in der angewandten Methode der Fehlerkorrektur.

Der Feldende-Editor sollte nur eingesetzt werden, wenn eine sofortige Fehlerkorrektur während der Erfassung möglich und notwendig ist. Das ist normalerweise dann gegeben, wenn die Daten unmittelbar am Ort ihres Entstehens auch erfaßt werden (Sachbearbeiterterminal).

Bei jeder Massendatenerfassung sollte jedoch der Stapelende-Editor vorgezogen werden, da es mit ihm möglich ist, eine Fehlerliste zu erstellen, die anschließend in den Fachabteilungen korrigiert wird.

Die Tatsache, daß der Stapelende-Editor die Datenerfassung nicht verzögert, ist ein wesentlicher Vorteil gegenüber dem Feldende-Editor. Da der Feldende-Editor während der Erfassung abläuft, geht seine Verarbeitungszeit voll in die Erfassungszeit ein.

Die folgende Tabelle zeigt die unterschiedlichen Methoden der Fehlersignalisierung:

Fehler-Anzeige	Programm-Befehl	Programm-Typ	Aktion
Fehlermeldung	PAUSE	Feld Stapel	Fehlerton, Programmunterbrechung, Fehleranzeige auf dem Bildschirm
Fehlerkennzeichen	FLAG	Feld Stapel	Setzt Fehlerkennzeichen auf angegebene Zeichen-/Feldposition
Fehlerliste auf Platte	OUTPUT	Stapel	Erstellt Fehlerliste auf der Magnetplatte
Fehlerliste auf Drucker	TYPE	Feld Stapel	Erstellt Fehlerliste auf dem Arbeitsplatzdrucker

Methoden der Fehlersignalisierung

Die nächste Tabelle zeigt die Unterschiede zwischen den Editoren auf:

	Feldende-Editor	Stapelende-Editor
Wann durchlaufen?	Im Eingabemodus nach jedem Feld, das Edit erlaubt.	In der Datenerfassungsebene bei BEENDEN eines Stapels
	Im Prüf-, Update- und Untersuchungsmodus wenn ein Feld verändert wird, das Edit erlaubt.	In der Supervisorebene nach Aufruf.
Fehleranzeige	PAUSE – Fehleranzeige FLAG – Fehlerkennzeichen TYPE – Fehlerliste auf dem Arbeitsplatzdrucker	PAUSE – Fehleranzeige FLAG – Fehlerkennzeichen TYPE – Fehlerliste auf dem Arbeitsplatzdrucker OUTPUT – Fehlerliste auf der Magnetplatte
Variable	Maximal 11. Werden nur bei Programmstart initialisiert	Maximal 99. Werden nur bei Programmstart initialisiert
Einsatz	Wenn sofortige Reaktion des Bedieners notwendig ist	Wenn die Fehlerkorrektur nach der Erfassung vorgenommen wird

Unterschiede zwischen Feldende- und Stapelende-Editor

	Programmstruktur

3.4 Ausgabeprogramm

Mit einem Ausgabeprogramm können Eingabedaten aus einem oder mehreren Stapeln reformatiert (verändert und umgestellt) und - ggf. unter Einfügung von neuen Daten - auf jedes angeschlossene Peripheriegerät (Drucker, Band, Leitung) ausgegeben werden.

Eine Reformatierung bei der Ausgabe kann aus folgenden Gründen sinnvoll sein:

- Bei der Erstellung des Eingabeformates hält man sich im Normalfall an die Reihenfolge der Daten des Ursprungsbelegs, während für die Ausgabe der Daten eine völlig andere Reihenfolge zweckmäßig sein kann.

- Im Fall der Reformatierung entfällt das Duplizieren und Überspringen von Feldern, wenn der Stapel auf die Platte geschrieben wird.
  Der Eingabe- und Prüfvorgang wird beschleunigt, und es wird Platz auf der Platte eingespart.

- Eine Ursprungsdatei kann für Ausgabedateien verschiedener Anwendungen benutzt werden.

- Während der Ausgabe können Daten aus Stammdateien eingefügt werden.

Die wichtigsten Funktionen, die in einem Ausgabeprogramm durchgeführt werden können, sind:

- Umstellen von Feldern innerhalb eines Satzes
- Verändern von Daten stellen- und feldweise
- Einfügen von Nullen, Blanks oder Sonderzeichen
- Einfügen von Daten und Satz-/Blockzählern
- Erstellen von Drucklisten mit Überschriften und Seitenzahlen
- Errechnen von neuen Daten durch arithmetische Operationen
- Einfügen von neuen Daten, basierend auf dem Inhalt eines anderen Feldes oder einer Rechenoperation
- Einfügen von Konstanten
- Einfügen von Stammdaten

Ablauf und Struktur des Ausgabeprogramms ist gleich dem Stapelende-Editor.

Programmstruktur

3.5 Sortierprogramm

Das Sortierprogramm erzeugt anhand bestimmter Kriterien eine Liste von Verweisen, mit deren Hilfe die Sätze eines oder mehrerer Stapel in aufsteigender oder absteigender Folge verkettet werden.

Während des Sortierlaufes sind - wie in jedem anderen Editorprogramm - Prüfungen, Berechnungen, Setzen von Fehlerkennzeichen und Datenübernahme aus Stammdateien möglich.

Die entstandene Liste von Verweisen kann anschließend als sortierter Stapel oder als Index einer Stammdatei verwendet werden.

Der Sortierprozess läuft in drei Phasen ab:

- Phase I

  Eine Liste der Sortierschlüssel mit Verweisen auf die zugehörigen Datensätze wird erzeugt.

- Phase II

  Die Liste wird auf- oder absteigend sortiert.

- Phase III

  Diese Phase kann in zwei Varianten durchlaufen

  - Indexdatei: Es werden höhere Indizes gebildet, um den Dateizugriff zu beschleunigen (siehe Punkt 4.1).

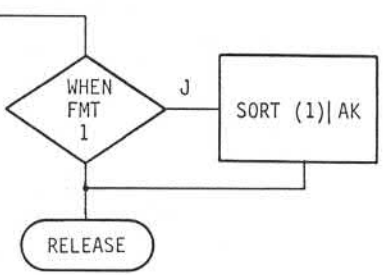
  - Sortierter Stapel: Die Sortierschlüssel sind jetzt irrelevant und werden aus der Liste entfernt

Das Eintragen der Sortierschlüssel in die in Phase I erstellte Liste geschieht durch den Befehl SORT. Es ist nicht zwingend notwendig, alle Sätze eines Stapels zu sortieren. Der SORT-Befehl kann von Bedingungen abhängig gemacht werden.

	Programmstruktur

Beispiel:

Schlüssel	FMT	Satz-Nr.
25142	1	1
71145	1	2
18501	2	3
62312	1	4
58711	2	5
37621	2	6
51248	1	7
65213	1	8



Unsortierter Stapel mit Sortierprogramm zur Sortierung aller Sätze mit Eingabeformat 1 in aufsteigende Reihenfolge.

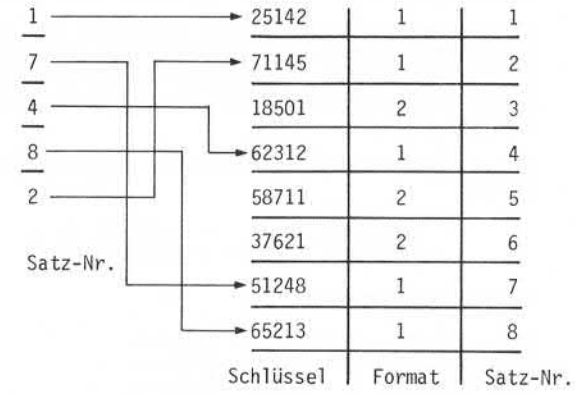
Schlüssel	Satz-Nr.
25142	1
71145	2
62312	4
51248	7
65213	8

Die in Phase I erstellte Liste der Schlüsselbegriffe mit Verweisen.

Programmstruktur

Schlüssel	Satz-Nr.
25142	1
51248	7
62312	4
65213	8
71145	2

Die sortierte Liste nach Ablauf von Phase II



Die sortierte Liste mit dem Ursprungsstapel nach Ablauf der Phase III.

	Programmstruktur

Die in Phase III entstandene Liste erhält einen max. 10stelligen Namen und kann anschließend wie ein normaler Stapel verarbeitet werden. In gleicher Weise ist es möglich, mehrere Stapel miteinander zu einem sortierten Stapel oder einer Indexdatei zu verbinden (siehe Multi-Batch-Verarbeitung im Stapelende-Editor).

Der verwendete Sortierschlüssel kann aus mehreren Einzelbegriffen bestehen. Jeder dieser Einzelbegriffe kann ein Feld, eine Variable, ein Literal oder eine Befehlsergänzung sein. Die Begriffe können in beliebiger Folge auftreten.

Beispiel:

Ein Zeitungsvertrieb führt zwei Stapel mit Kundenstammdaten. Der erste Stapel enthält alle Kunden, die Tageszeitungen beziehen, der zweite Stapel alle Kunden, die Illustrierte beziehen.

Die vorhandenen 8 Stadtbezirke sollen auf zwei Vertreter aufgeteilt werden, wobei für jeden Vertreter zwischen Zeitungs- und Illustriertenkunden unterschieden werden soll. Innerhalb jeder Gruppe werden die Kunden nach Straße und Hausnummer aufsteigend sortiert.

Die Zuordnung der Stadtbezirke soll folgendermaßen vorgenommen werden:

Vertreter Maier - Bezirk 2, 3, 6
Vertreter Schulze - Bezirk 1, 4, 5

Der Kundenstammsatz:

Name	Straße	Hausnummer	Bezirk
Feld 1	Feld 2	Feld 3	Feld 4

Programmstruktur

Das zugehörige Sortierprogramm:

```
IF ␣ (4)␣=␣2␣OR␣=␣3␣OR␣=␣6

     SORT␣'MAIER'␣ BATCH ␣ (2)␣(3).

IF ␣ (4)␣=␣1␣OR␣=␣4␣OR␣=␣5

     SORT␣'SCHULZE'␣ BATCH ␣ (2)␣(3).

RELEASE.
```

Der Datenstapel "///ILLUSTR"

Name	Strasse	Hausnummer	Bezirk
Glesner	Kurfuerstenstr.	66	1
Frerichs	Platte Str.	16	2
Stolze	Leuschnerstr.	11	3
Ramberg	Platte Str.	05	2
Gluecklich	Nahestr.	22	5
Koenig	Kurfuerstenstr.	13	1

Der Datenstapel "///ZEITUNG"

Name	Strasse	Hausnummer	Bezirk
Abele	Talstr.	15	4
Bendix	Auf dem Meere	55	5
Woop	Kreuzbergstr.	15	3
Dienstknecht	Kreuzbergstr.	02	3
Roelfke	Lagesche Str.	36	2
Naumes	Lagesche Str.	31	2
Leineweber	Hauptstr.	18	1
Klapper	Pfeiffergasse	25	6
Paul	Breslauerstr.	16	3

Programmstruktur

Aus beiden Stapeln wird die folgende Liste der Sortierschlüssel gebildet:

SCHULZE	///ILLUSTR	Talstr.	15
SCHULZE	///ILLUSTR	Auf dem Meere	55
MAIER	///ILLUSTR	Kreuzbergstr.	15
MAIER	///ILLUSTR	Kreuzbergstr.	02
MAIER	///ILLUSTR	Lagesche Str.	36
MAIER	///ILLUSTR	Lagesche Str.	31
SCHULZE	///ILLUSTR	Hauptstr.	18
MAIER	///ILLUSTR	Pfeiffergasse	25
MAIER	///ILLUSTR	Breslauerstr.	16
SCHULZE	///ZEITUNG	Kurfürstenstr.	66
MAIER	///ZEITUNG	Platte Str.	16
MAIER	///ZEITUNG	Leuschnerstr.	11
MAIER	///ZEITUNG	Platte Str.	05
SCHULZE	///ZEITUNG	Nahestr.	22
SCHULZE	///ZEITUNG	Kurfürstenstr.	13
Ver-treter	Stapelname	Strasse	Hausnummer

Programmstruktur

Der Stapel nach abgeschlossener Sortierung:

Maier	Paul	Breslauerstr.	163	Illustrierte
	Dienstknecht	Kreuzbergstr.	023	
	Woop	Kreuzbergstr.	153	
	Naumes	Lagesche Str.	312	
	Roelfke	Lagesche Str.	362	
	Klapper	Pfeiffergasse	256	
	Stolze	Leuschnerstr.	113	Zeitung
	Ramberg	Platte Str.	052	
	Frerichs	Platte Str.	162	
Schulze	Bendix	Auf dem Meere	555	Illustrierte
	Leineweber	Hauptstr.	181	
	Abele	Talstr.	154	
	Koenig	Kurfuerstenstr.	131	Zeitung
	Glesner	Kurfuerstenstr.	661	
	Gluecklich	Nahestr.	225	

Treten in einem Datenstapel Sortierbegriffe mit identischem Inhalt auf, kann durch eine Positionsangabe im SORT-Befehl für diese Sortierbegriffe eine Reihenfolge vorgegeben werden.

Beispiel:

Auf der Magnetplatte befindet sich ein Stapel, der die Daten einer Kundendatei enthält.

Bei der täglichen Erfassung werden nur Kundennummern mit den entsprechenden Bewegungen in einem Stapel erfaßt.

Bei der Ausgabe sollen jedoch die Stammdaten des Kunden gemeinsam mit den Bewegungen ausgegeben werden.
Aus diesem Grund wird der Stapel mit den Stammdaten und der Stapel mit den Bewegungsdaten sortiert.

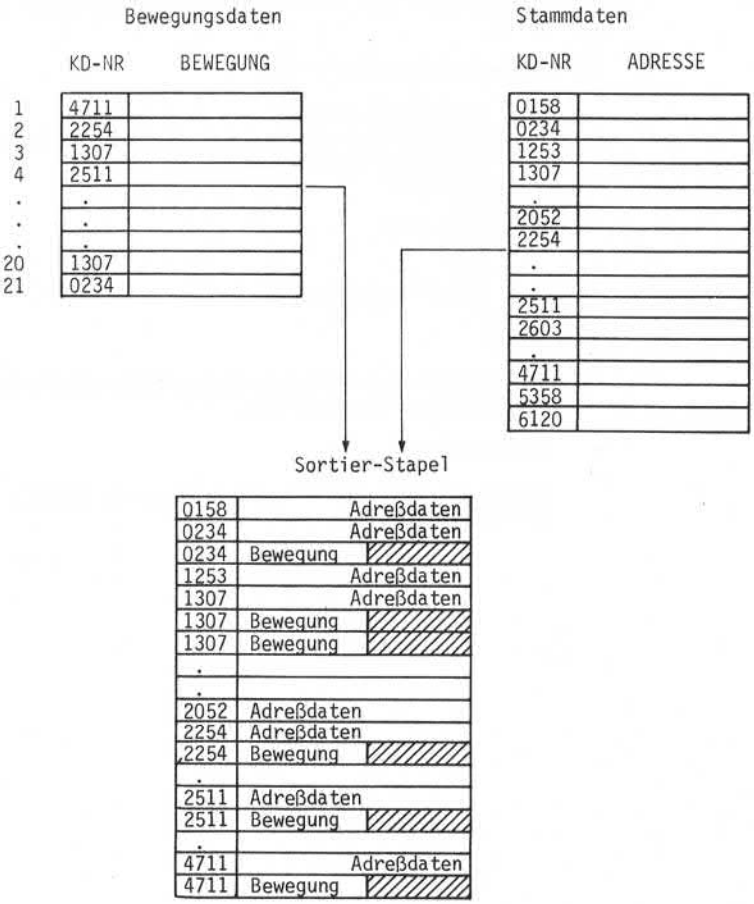
	Programmstruktur

Um ein Unterscheidungsmerkmal zu haben, wurden die Bewegungsdaten mit FMT1, die Stammdaten mit FMT2 erfaßt.

Das Sortierprogramm sieht über den Positionszähler vor, daß Stammdaten im sortierten Stapel immer vor den Bewegungsdaten stehen.

```
WHEN␣FMT␣1␣SORT␣(1)'2'.
WHEN␣FMT␣2␣SORT␣(1)'1'.
RELEASE.
```

	Programmstruktur



	Programmstruktur

Die Reihenfolge der Sortierbegriffe wird durch die Vergleichsfolge festgelegt. Drei verschiedene Vergleichsfolgen stehen zur Verfügung.

- Alphanumerisch (Radix 40)

  - Leerzeichen (Blank)
  - Sonderzeichen (keine Unterscheidung zwischen verschiedenen Sonderzeichen)
  - Plus Null
  - A-I
  - Minus Null
  - J-Z
  - 0-9
  - Fehlerkennzeichen

- EBCDIC

  Im Gegensatz zur alphanumerischen Folge wird zwischen Sonderzeichen unterschieden.

- Numerisch 0-9

  Belegt nur halb soviel Speicherplatz wie EBCDIC, daher sehr effektiv.

Aufbau und Verwaltung von Indexdateien

4. Aufbau und Verwaltung von Indexdateien

Das System 620/45 bietet durch die systemunterstützte Verwaltung von Indexdateien die Möglichkeiten einer komfortablen Dateiführung.

Die Vorteile der Dateiverwaltung des Systems sind:

- Es können unbegrenzt viele Dateien auf dem System geführt werden.
- Je Programm können bis zu 64 Dateien angesprochen werden.
- Maximal 4 Dateien können je Programm zur gleichen Zeit eröffnet sein.
- Die Anzahl der Schlüsselfelder ist nicht begrenzt.
- Die Position der Schlüsselfelder innerhalb des Dateisatzes ist nicht vorgeschrieben.
- Jeder Dateisatz kann maximal 255 Indizes zugeordnet werden.
- Die Schlüssellänge beträgt maximal 48 Zeichen.
- Der Schlüsselbegriff muß nicht unbedingt im Dateisatz enthalten sein. Er kann auch durch eine arithmetische Operation errechnet werden.
- Dateisätze können eingefügt und gelöscht werden. Die eingefügten Sätze sind sofort danach - ohne Reorganisationslauf - im Direktzugriff.
- Auf die Daten kann aus allen Programmstufen index-sequentiell zugegriffen werden.

Aufbau und Verwaltung von Indexdateien

Jede Indexdatei besteht aus einem Datenstapel und dem zugehörigen Index. Während der Erstellung einer Indexdatei werden die aus dem Datenstapel entnommenen Schlüsselbegriffe sortiert und im Index abgestellt. Gleichzeitig werden Schlüsselbegriff und Datensatz über ein Zeigerfeld verknüpft. Der Datenstapel bleibt unverändert.

Der Index enthält einen max. 8stelligen Namen, der aus beliebigen Tastaturzeichen bestehen kann. Lediglich das erste Zeichen muß ein Buchstabe sein. Unter diesem Namen ist der Index und sein zugehöriger Datenstapel jederzeit als Indexdatei ansprechbar.

4.1 Indexaufbau

Der Index wird durch die Supervisorfunktion DATEIVERWALTUNG mit Hilfe eines vom Anwender geschriebenen Sortierprogramms aufgebaut.

Der Prozess läuft in drei Phasen ab:

- Phase I

  Die vom Anwender im SORT-Befehl spezifizierten Schlüsselbegriffe werden in den Index eingetragen und mit den zugehörigen Datensätzen verknüpft.

- Phase II

  Der Index wird sortiert.

- Phase III

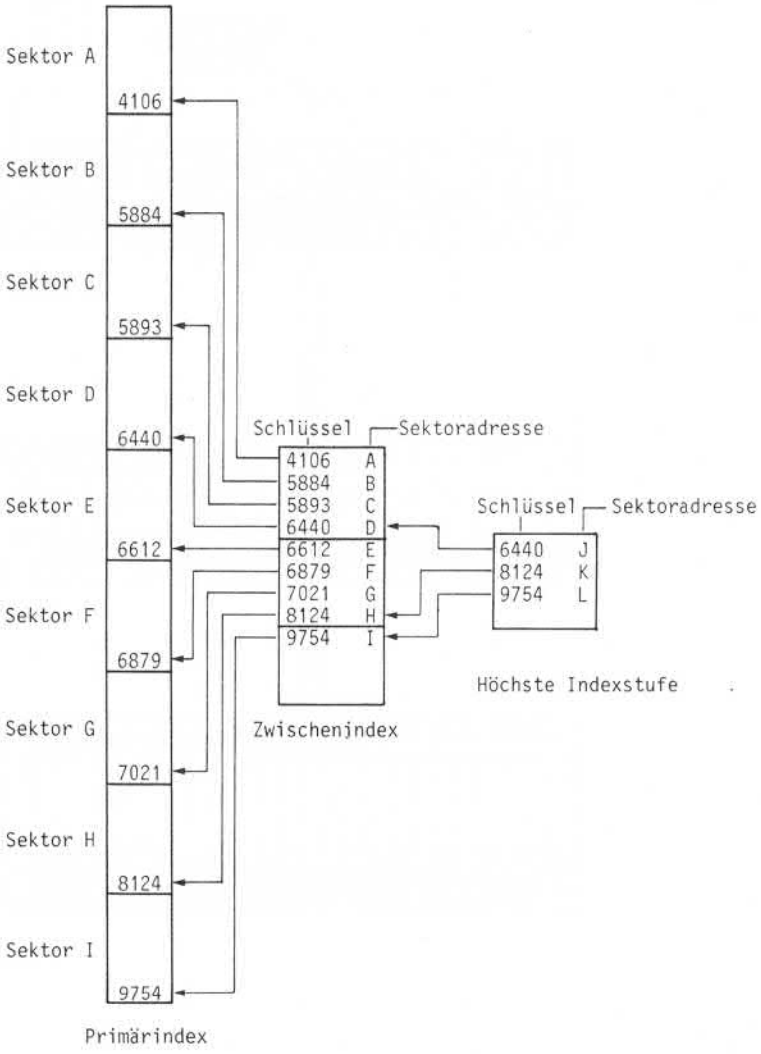
  Zusätzlich zum Primärindex werden höhere Indizes gebildet, um den Zugriff auf die Datensätze zu beschleunigen.

Aufbau und Verwaltung von Indexdateien

Beispiel:

	Aufbau und Verwaltung von Indexdateien

Die in Phase III gebildeten höheren Indizes enthalten die Sektorenadressen des vorhergehenden Index und den höchsten Schlüsselbegriff jedes Sektors.



Aufbau und Verwaltung von Indexdateien

In Phase III werden solange höhere Indizes gebildet, bis eine Indexstufe erreicht ist, die einen oder zwei Sektoren belegt. Dies ist die höchste Indexstufe. Jeder Dateizugriff beginnt mit der höchsten Indexstufe und führt über alle niedrigeren Indexstufen bis zum Primärindex.

Die Form, in der die Schlüsselinformation im Index gespeichert wird, bestimmt der Schlüsseltyp:

- N - nur numerisch

  Der Schlüssel wird in gepackter Darstellung gespeichert. 2 Ziffern belegen 1 Byte.

- A - alphanumerisch (Radix 40)

  Der Schlüssel wird in komprimierter Form gespeichert. 3 Zeichen belegen 2 Byte. Nur die Buchstaben A-Z, Blank und die Ziffern 0-9 sind zulässig.

- E - EBCDIC

  Der Schlüssel wird im 8-Bit Format gespeichert. 1 Zeichen belegt 1 Byte. Alle Zeichen sind zulässig.

Folgende Tabelle gibt Aufschluß über Schlüsseltyp, zulässige Zeichen und Schlüssellänge:

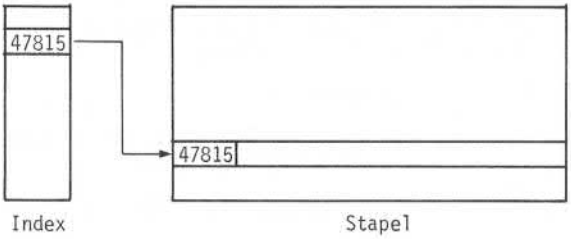
	Leerz.	A-Z	0-9	Sonderzeichen	max. Länge
N	N	N	J	N	48
A	J	J	J	N	36
E	J	J	J	J	24

Der Aufbau von Schlüsselbegriffen und Index wird durch das vom Anwender geschriebene Sortierprogramm bestimmt. Daher kann jeder Anwender die für ihn optimale Dateiorganisation programmieren. In den folgenden Abschnitten werden die möglichen Formen der Indexorganisation vorgestellt.

Aufbau und Verwaltung von Indexdateien

4.1.1 Ein Index für eine Datei, ein Schlüssel pro Satz

Dies ist die einfachste Form der Indexorganisation. Für jeden Satz eines Datenstapels wird ein Eintrag im Index vorgenommen. Jeder Satz kann direkt durch seinen Schlüssel adressiert werden.

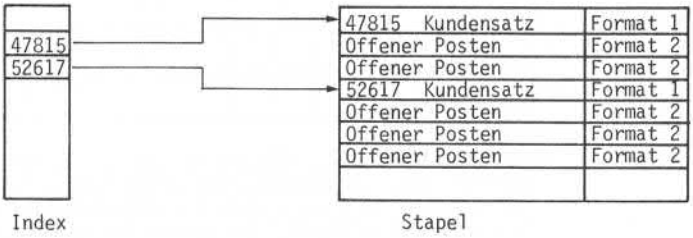


Das zugehörige Sortierprogramm:

```
SORT␣(1).

RELEASE.
```

4.1.2 Ein Index für eine Datei, ein Schlüssel für mehrere Sätze

Bei dieser Form der Indexorganisation wird nur für bestimmte Sätze innerhalb des Stapels ein Eintrag im Index vorgenommen. Auf Sätze, die nicht im Index eingetragen sind, muß durch FORWARD- oder BACK-Instruktionen zugegriffen werden.



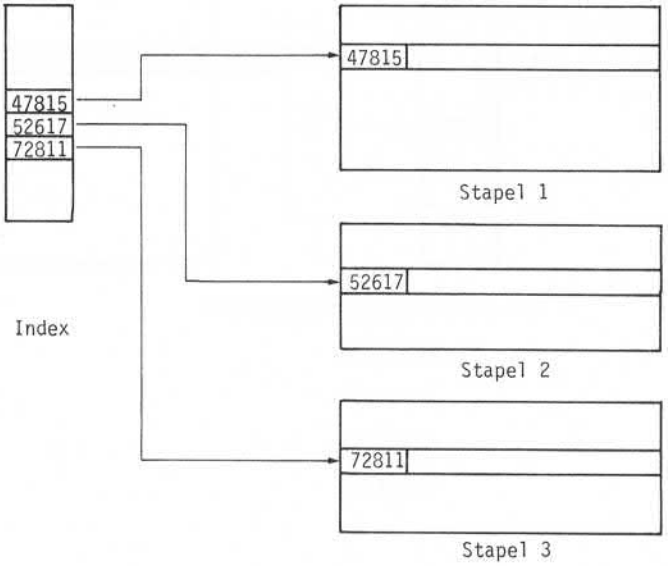
Das zugehörige Sortierprogramm:

```
WHEN FMT 1
     SORT (1).
RELEASE.
```

Aufbau und Verwaltung von Indexdateien

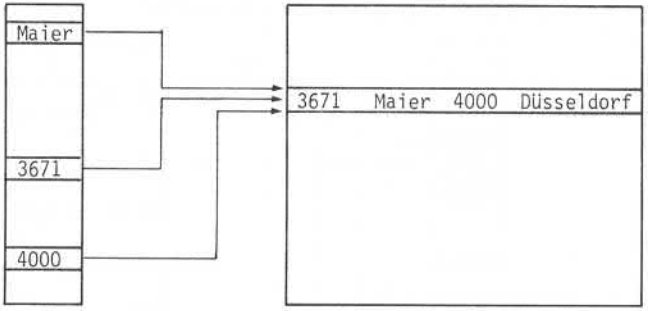
4.1.3 Ein Index für mehrere Datenstapel

Mehrere Datenstapel können durch Multi-Batch-Verarbeitung im Sortierprogramm zu einer Indexdatei zusammengefaßt werden. Beim Zugriff über den Index spielt es keine Rolle, in welchem Datenstapel sich der gesuchte Satz befindet.



4.1.4 Mehrere Schlüssel für einen Datensatz

Innerhalb eines Index können mehrere unterschiedliche Schlüssel auf einem Datensatz verweisen.



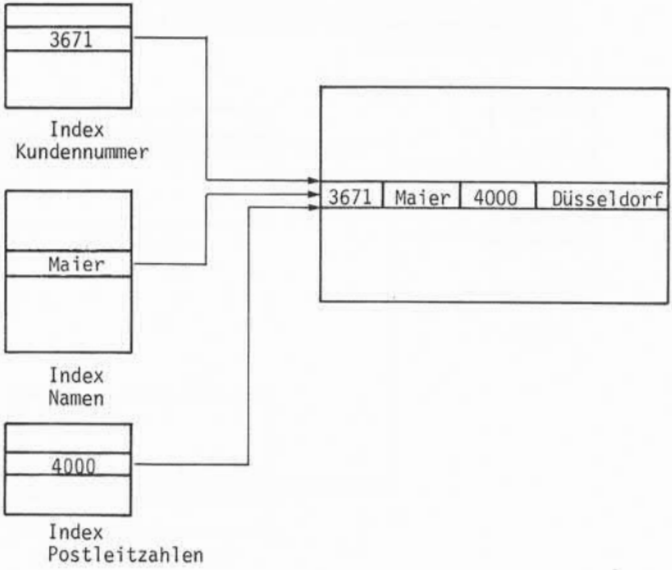
Das zugehörige Sortierprogramm:

```
SORT␣(1).
SORT␣(2).
SORT␣(3).
RELEASE.
```

Aufbau und Verwaltung von Indexdateien

4.1.5 Mehrere Indizes für einen Datenstapel

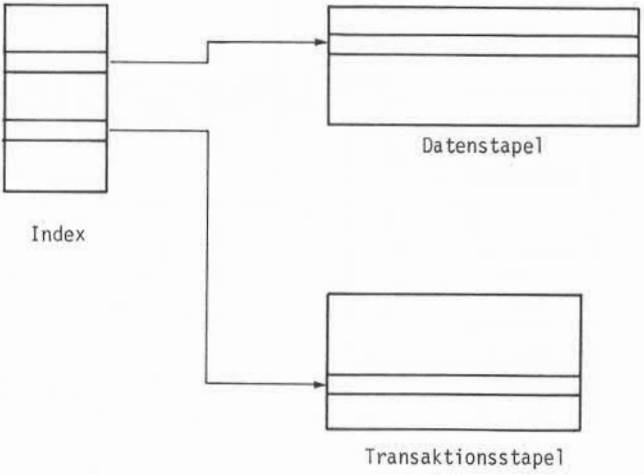
Auf jeden Datensatz innerhalb eines Stapels können bis zu 255 Schlüssel verweisen. Diese Schlüssel können auf bis zu 255 Indizes verteilt werden.



Jeder Index muß mit einem eigenen Sortierprogramm aufgebaut werden. Aus der Sicht des Anwenderprogrammes repräsentiert jeder Index eine unabhängige Datei.

4.1.6 Verknüpfung des Transaktionsstapels mit der Indexdatei

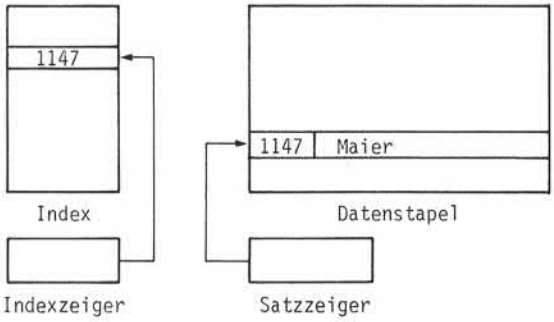
Soll während eines Feldende- oder Stapelende-Editor-Laufes Datensätze des Transaktionsstapels mit dem Index verknüpft werden, muß die Indexdatei mit dem Befehl OPEN|INC eröffnet werden. Nach Beendigung des Editor-Laufes kann der Transaktionsstapel nicht mehr gelöscht werden, solange die Indexdatei besteht.



Aufbau und Verwaltung von Indexdateien

4.2 Zugriffsmethode

In jedem Programm, das auf Indexdateien zugreift, werden zwei Zeigerfelder geführt. Ein Zeiger verweist auf den zuletzt gelesenen Schlüssel im Index (Indexzeiger), der andere Zeiger auf den zuletzt gelesenen Satz im zugehörigen Datenstapel (Satzzeiger).



Beide Zeiger werden abhängig vom jeweiligen Datei-Zugriff verändert.

Besonders zu beachten ist die Tatsache, daß zwar 4 Indexdateien gleichzeitig in einem Programm eröffnet werden können, für alle Dateien aber nur je ein Zeiger zur Verfügung steht. Das bedeutet, daß immer nur ein Schlüssel und ein Datensatz zu einem gegebenen Zeitpunkt im Zugriff ist. Jeder Dateizugriff läuft daher in zwei Stufen ab: Zurückschreiben eines veränderten Satzes und anschließend Lesen des aktuellen Satzes.

Wird beim Dateizugriff der gewünschte Satz nicht gefunden, ist anschließend der Inhalt beider Zeiger zerstört.

Zwei grundsätzlich unterschiedliche Arten des Dateizugriffs sind möglich: Zugriff über den Index oder direkter Zugriff auf den Datenstapel

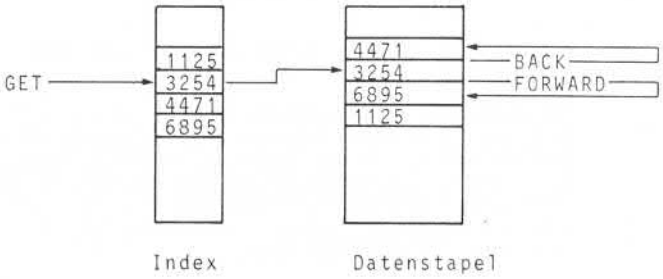
Aufbau und Verwaltung von Indexdateien

Der Zugriff über den Index wird durch die Instruktion GET ausgeführt. GET verändert den Indexzeiger und den Satzzeiger.

Für den index-sequentiellen Zugriff stehen die Befehle GET FIRST (lese ersten Satz im Index), GET LAST (lese letzten Satz im Index), GET PRIOR (lese vorhergehenden Satz im Index) und GET NEXT (lese nächsten Satz im Index) zur Verfügung.

Soll sequentiell auf die Sätze des Datenstapels zugegriffen werden, stehen die Befehle FORWARD (lese nächsten Satz im Stapel) und BACK (lese vorhergehenden Satz im Stapel) zur Verfügung. Beide Befehle verändern nur den Satzzeiger.

Besonders zu beachten ist, daß beim sequentiellen Lesen im Datenstapel nicht die logische (durch den Index vorgegebene), sondern die physikalische Reihenfolge der Sätze im Datenstapel eingehalten wird.



Da durch FORWARD und BACK nur der Satzzeiger verändert wird, kann anschließend durch die Instruktion GET CURRENT wieder auf den im Indexzeiger angegebenen Satz zugegriffen werden.

Aufbau und Verwaltung von Indexdateien

4.3 Verwaltung

Zur Verwaltung einer Indexdatei sind folgende Funktionen verfügbar:

- Datensätze lesen, verändern und zurückschreiben
- Einfügen von Sätzen in den Datenstapel
- Einfügen von Schlüsseln in den Index
- Löschen von Sätzen im Datenstapel
- Löschen von Schlüsseln im Index

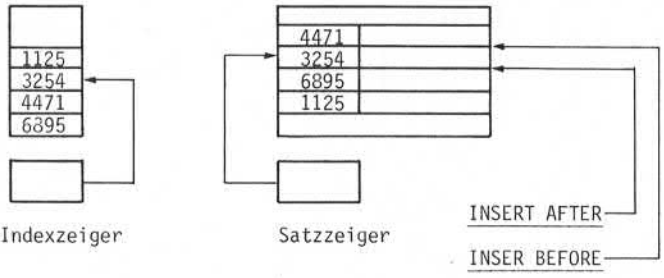
4.3.1 Datensätze lesen, verändern und zurückschreiben

Die zu verändernden Datensätze werden entweder durch die Instruktion GET oder durch die Instruktion FORWARD/BACK gelesen. Anschließend können die gelesenen Daten vom Anwenderprogramm verändert werden. Das Zurückschreiben wird vom Betriebssystem durchgeführt, sobald ein neuer Satz gelesen oder das Programm beendet wird.

4.3.2 Einfügen von Sätzen in den Datenstapel

Abhängig vom Stand des Satzzeigers kann eine Leersatz in den Datenstapel eingefügt werden. Der Satz wird entweder vor dem zuletzt gelesenen Satz (INSERT BEFORE) oder hinter dem zuletzt gelesenen Satz (INSERT AFTER) eingefügt. In beiden Fällen wird der Satz in die physikalische Folge der Sätze im Datenstapel eingefügt, die von der durch den Index vorgegebenen logischen Folge abweichen kann. Ist der Satz eingefügt, kann er vom Anwenderprogramm mit Daten gefüllt werden. Das anschließende Zurückschreiben auf die Platte wird vom Betriebssystem durchgeführt.

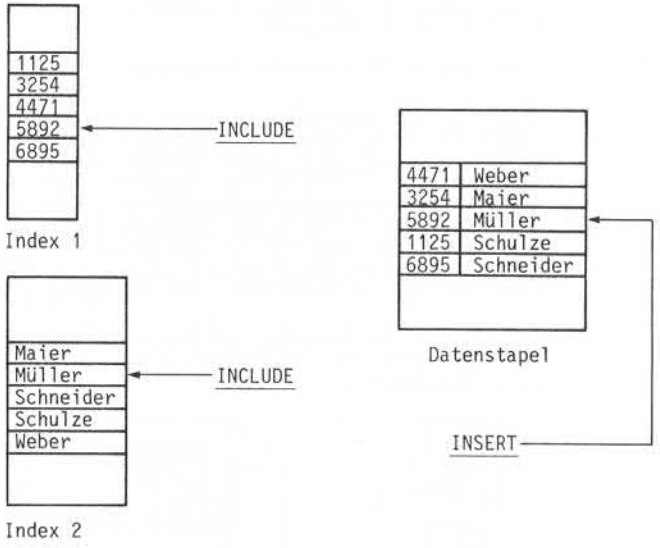
Besonders zu beachten ist, daß der eingefügte Satz nicht mit dem Index verkettet ist. Das Verketten geschieht erst durch die Instruktion INCLUDE.



4.3.3 Einfügen von Schlüsseln in den Index

Ein Satz, der in den Datenstapel aufgenommen wurde, wird durch Einfügen seines Schlüssels mit dem Index verkettet. Das Einfügen geschieht durch den Befehl INCLUDE. Der Schlüssel wird an der durch die logische Folge im Index vorgegebenen Stelle eingefügt.

Es können außer den Sätzen des Datenstapels auch Sätze des Transaktionsstapels mit dem Index verkettet werden. Voraussetzung dafür ist, daß die Datei durch den Befehl OPEN|INC eröffnet wurde. Jeder neue Datensatz kann mit bis zu 255 Indizes verkettet werden. Zu diesem Zweck müssen nach dem INSERT-Befehl nur die entsprechenden INCLUDE-Befehle codiert werden.



Aufbau und Verwaltung von Indexdateien

4.3.4 Löschen von Sätzen im Datenstapel

Ein Satz im Datenstapel wird durch die Instruktion DELETE gelöscht. Vor Ausführung des Befehls muß sichergestellt werden, daß der Satzzeiger auf den zu löschenden Satz verweist.

Bestehende Verkettungen des Satzes mit Indizes werden automatisch durch das Betriebssystem gelöscht (auch wenn der Satz manuell gelöscht wird).

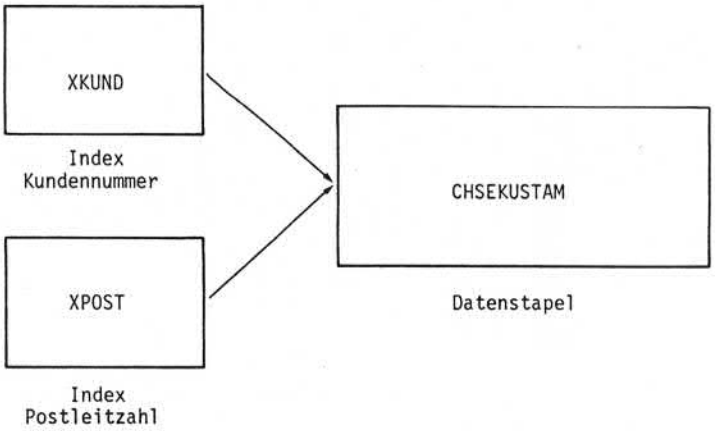
4.3.5 Löschen von Schlüsseln im Index

Durch die Instruktion REMOVE wird der durch den INDEXZEIGER angegebene Schlüssel im Index gelöscht. Das Löschen von Schlüsseln ist nur erforderlich, wenn die Verkettung eines ungelöschten Satzes mit einem Index aufgehoben werden soll.

4.3.6 Programmbeispiel

Auf den folgenden Seiten finden Sie als Beispiel das Verwaltungsprogramm für eine Kundenstammdatei. Für diese Kundenstammdatei existieren zwei Indizes. Ein Index erlaubt den Zugriff über die Kundennummer, der zweite Index über die Postleitzahl.

Mit Hilfe des Programms können Sätze verändert, eingefügt oder gelöscht werden.



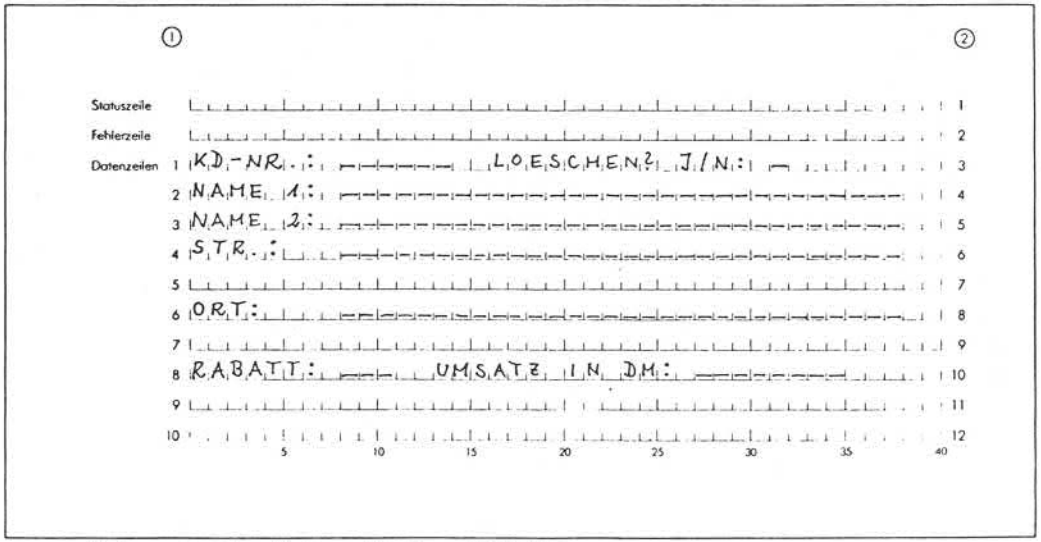
System 620

Display-Aufbau 480/360 Zeichen

Firma _____

Organisator _____ Name des Eingabe-Formats CHSEKUND Datum _____

Seite: _____ von: _____



① - Zeilennummer f. Ein/ausFormat ② - Zeilennummer f. Editor Befehle wie ACCEPT/PAUSE/SHOW

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name K U S T A M M Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit ≠ beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Kon.-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füll-zeich: Mit Daten	Füll-zeich: Ohne Daten	Stap. Sum.: Add./Sub.	Stap. Sum.: Nummer	Tabelle: Nummer	Tabelle: Urg. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-Funktion	Prüf-Ziffer: Nummer	Prüf-Ziffer: Methode	Prüf-Ziffer: Matric. Felder	Update	Aufsteig. Folge	Tab.-Feld	Zusatz-Feld	Kettung	Feldende-Echar	Feld einf. /löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	23	24	25	26	27	28	29	30	31	32	33
0	1	1	KUNDEN-STAMMDATEN																											
1	3	1	KD-NR.: ≠	6	N			Y	Y																				Y	
2	3	17	LOESCHEN? J/N: ≠	1	L																								Y	
3	4	1	NAME 1: ≠	30	L																									
4	5	1	NAME 2: ≠	30	L																									
5	6	1	STR.: ≠	30	L																									
6	8	1	ORT: ≠	30	L																									
7	9	1	RABATT: ≠	3	N						Y																			
8	9	14	UMSATZ IN DM: ≠	8	N						Y																		Y	

Feld-Nr.		21	Min-Wert		22	Max-Wert

```
DECLARE SCHL.                          Deklarieren der Variablen
DEFINE XKUND XPOST.                    Definieren der Dateien
WHEN START
     OPEN XKUND|UPD; OPEN XPOST|UPD.   Dateien eröffnen
WHEN NOT FIELD 1                       Feld 1 = Kundennummer
     GOTO !A100.
GET XKUND USING (1)                    Kundenstammsatz lesen
     ELSE GOTO !A200.                  Stammsatz nicht vorhanden
POSITION (2).                          Zurück zur Abfrage LOESCHEN? J/N: im Eingabeformat
!A100
WHEN NOT FIELD 2 GOTO !A300.           Feld 2 = LOESCHEN? J/N
IF (2) ≠ 'J' GOTO !A150.               J = Satz löschen
DELETE XKUND.                          Satz im Datenstapel löschen
POSITION (1).                          Zurück zur Kundennummer
!A150
IF (2) ≠ 'N' POSITION (2).             Falsche Eingabe in Feld 2
MOVE à3â TO (3).                       Daten des Stammsatzes in den Transaktionsstapel
                                       übertragen
MOVE à4â TO (4).
MOVE à5â TO (5).
MOVE à6â TO (6).
```

```
MOVE ä7ä TO (7).
MOVE ä8ä TO (8).
POSITION (3).                          Verzweigen auf das    Datenfeld des Transaktionsstapels
!A200
 MOVE 'I' TO (2).                      Kennzeichen f. eingefügten Satz
POSITION (3).                          Verzweigen auf das    Datenfeld des Transaktionsstapels
!A300
 IF (2) ≠ 'I'                          I = Satz einfügen
     GOTO !A400.
INSERT 1 AFTER XKUND.                  Leersatz einfügen
!A400
MOVE (1) TO ä1ä.                       Daten des Transaktionsstapels in den Stammsatz übertra-
                                       gen
MOVE (2) TO ä2ä.
MOVE (3) TO ä3ä.
MOVE (4) TO ä4ä.
MOVE (5) TO ä5ä.
MOVE (6) TO ä6ä.
MOVE (7) TO ä7ä.
MOVE (8) TO ä8ä.
IF (2) = 'I'                           I = Satz einfügen
```

Aufbau und Verwaltung von Indexdateien

```
INCLUDE XKUND IN XKUND USING @1@;
MOVE @6:1-4@ TO SCHL;
   INCLUDE XKUND IN XPOST USING SCHL.
RELEASE.
```

Kundennummer in den Index Kundennummer einfügen
Postleitzahl → SCHL
Postleitzahl in den Index Postleitzahl einfügen.

4.4 Reorganisation

Abhängig von der Änderungshäufigkeit sollte eine Indexdatei in regelmäßigen Abständen reorganisiert werden.

Zwei Arten der Reorganisation sind möglich:

- Reorganisation des Index

  Zu diesem Zweck wird der Index gelöscht und anschließend neu aufgebaut. Diese Reorganisation verringert den Platzbedarf des Index.

- Reorganisation von Datenbereich und Index

  Der Datenstapel wird auf Band ausgegeben, der Index gelöscht und nach Einlesen des Datenstapels neu aufgebaut. Diese Reorganisation verringert den Platzbedarf des Datenstapels und des Index.

5. Befehlsstruktur

5.1 Befehlstypen

Abhängig von der durchzuführenden Operation läßt sich der Befehlsvorrat des Systems in 8 Gruppen gliedern:

- Arithmetik und Datentransport
- Arbeitsplatzsteuerung
- Stapelverarbeitung
- Verarbeitung von Indexdateien
- Programmsteuerung
- Datenausgabe
- Sortieren
- Sonderbefehle

5.1.1 Arithmetik und Datentransport

Zu dieser Gruppe gehören folgende Befehle:

Befehl	Funktion	Seite
DECLARE	Deklarieren von Variablen	123
MOVE	Datenübertragung zwischen zwei Speicherbereichen	159
ADD	Addition	107
SUBTRACT	Subtraktion	207
MULTIPLY	Multiplikation	159
DIVIDE	Division	129

5.1.2 Arbeitsplatzsteuerung

Mit Hilfe der Befehle zur Arbeitsplatzsteuerung können Daten auf dem Bildschirm angezeigt oder über die Tastatur eingegeben werden.

Befehl	Funktion	Seite
PAUSE	Anzeige einer Meldung an beliebiger Stelle des Bildschirms mit zusätzlichem Fehlerton und Programmhalt.	183
SHOW	Anzeige einer Meldung an beliebiger Stelle des Bildschirms.	201
ACCEPT	Dateneingabe in eine Variable erlauben.	105

5.1.3 Stapelverarbeitung

Befehle zur Stapelverarbeitung steuern den Zugriff der Anwenderprogramme auf Felder und/oder Sätze der Datenstapel.

Befehl	Funktion	Seite
RELEASE	Verlassen des Programms.	195
POSITION	Positionieren auf ein bestimmtes Feld des Eingabeformats.	189
BACK	Zugriff auf den vorhergehenden Satz des Datenstapels.	113
FORWARD	Zugriff auf den folgenden Satz des Datenstapels.	133
BYPASS	Rest des Stapels übergehen.	117
LINK	Bestimmtes Eingabeformat anwählen.	155

5.1.4 Verarbeitung von Indexdateien

Dem programmierten Zugriff und der Verwaltung von Indexdateien dienen folgende Befehle:

Befehl	Funktion	Seite
DEFINE	Definieren einer Datei.	125
OPEN	Eröffnen einer Datei zur Verarbeitung.	165
CLOSE	Schließen einer eröffneten Datei.	121
GET	Im Index eingetragenen Satz lesen.	139
FORWARD	Nächsten Satz im Datenstapel lesen.	135
BACK	Vorhergehenden Satz im Datenstapel lesen.	115
INSERT	Satz in den Datenstapel einer Indexdatei einfügen.	153
DELETE	Satz im Datenstapel einer Indexdatei löschen.	127
INCLUDE	Schlüssel in den Index einfügen.	149
REMOVE	Schlüssel im Index löschen.	197
PROTECT	Satz gegen den Zugriff eines anderen Anwenderprogramms schützen.	193

5.1.5 Programmsteuerung

Der logische Ablauf der Programme wird durch folgende Befehle gesteuert:

Befehl	Funktion	Seite
IF	Abfrage einer Bedingung.	143
WHEN	Abfrage einer Systembedingung.	211

Befehlsstruktur

Befehl	Funktion	Seite
GOTO	Unbedingte Verzweigung.	141
PERFORM	Verzweigung in eine Unterroutine.	185
EXIT	Ende einer Unterroutine, Rücksprung ins Hauptprogramm.	185
STOP	Abbruch des Programms.	205

5.1.6 Datenausgabe

Zur Datenausgabe auf Magnetband, Diskette, Platte, Systemdrucker, Arbeitsplatzdrucker und DFÜ stehen drei Befehle zur Verfügung:

Befehl	Funktion	Seite
AUDIT	Ausgabe auf Magnetband parallel zur Datenerfassung.	109
OUTPUT	Datenausgabe auf Magnetband, Diskette, Platte, Systemdrucker oder DFÜ.	167
TYPE	Datenausgabe auf den Arbeitsplatz-drucker.	209

5.1.7 Sortieren

Zum Sortieren von Datenstapeln oder zur Erstellung von Index-dateien steht der Befehl SORT (Seite 203) zur Verfügung.

5.1.8 Sonderbefehle

Befehl	Funktion	Seite
FLAG	Fehlerkennzeichen setzen.	129
CLEAR	Fehlerkennzeichen löschen.	119

Befehlsstruktur

Befehl	Funktion	Seite
SET	Inhalt einer Variablen in einen Stapelsummenakkumulator übertragen.	199
LOAD	Inhalt eines Stapelsummenakkumulators in eine Variable übertragen.	157
NOTE	Kommentar einfügen.	161

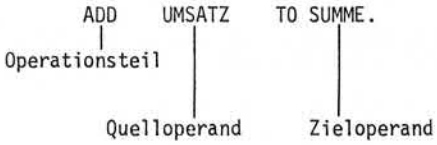
Befehlsstruktur

5.2 Operanden

In der Regel besteht ein Programmbefehl des Systems aus einem Operationsteil und einem oder mehreren Operanden. Der Operationsteil bestimmt die auszuführende Operation, die Operanden definieren die Datenbereiche mit denen die Operation ausgeführt werden soll.

Beispiel:

Addition zweier Zahlen



Die Operanden UMSATZ und SUMME sind die symbolischen Adressen zweier Bereiche im Systemspeicher.

Im System existieren folgende Operandentypen:

•	Feld	Der Inhalt eines Feldes in einem Stapel oder einer Indexdatei.
•	Variable	Ein Bereich innerhalb des Systemspeichers.
•	Numerisches Literal	Ein ganzzahliger, numerischer Wert.
•	Alphanum. Literal	Eine Zeichenkette
•	Befehls-ergänzung	Bestimmte Systemdaten

	Befehlsstruktur

5.2.1 Feld

Das Feld ist Teil eines Satzes in einem Datenstapel oder einer Indexdatei.

Ein Feld wieder durch Angabe seiner Feldnummer im Eingabeformat adressiert.

Befindet sich das Feld im Satz eines Datenstapels, wird die Feldnummer in Klammern eingeschlossen.

Beispiel: (5)
(318)

Wird ein Feld adressiert, das sich im Satz einer Indexdatei befindet, wird die Feldnummer in ə-Zeichen eingeschlossen.

Beispiel: ə5ə
ə318ə

5.2.2 Teilfeld

Durch die Adressierung von Teilfeldern können einzelne Zeichen oder Zeichenfolgen innerhalb eines Feldes verarbeitet werden.

Ein Teilfeld wird folgendermaßen adressiert:

(n:s-e) oder ən:s-eə

n = Feldnummer

s = Startposition des Teilfeldes innerhalb des Feldes

e = Endeposition des Teilfeldes innerhalb des Feldes

Die Positionsangaben werden von links nach rechts gezählt. Wird nur ein Zeichen adressiert, ist der Adressausdruck (n:s) oder ən:sə ausreichend.

Beispiel:

Inhalt Feld (3) 4790FRADERBORN

Adressierung der Postleitzahl

(3:1-4)

Adressierung der letzten Stelle der Postleitzahl

(3:4)

5.2.3 Indiziertes Feld

Unter Indizierung ist die indirekte Adressierung von Feldern oder Teilfeldern zu verstehen. Jedes Element des Adressausdrucks (n:s-e) oder @n:s-e@ kann durch eine Variable ersetzt werden. Die Inhalte dieser Variablen werden als Feldnummer bzw. Zeichenpositionen innerhalb des Feldes interpretiert.

Beispiel:

Der Inhalt der Variablen A ist = 7.

(A) = Feld 7

(2:A) = Feld 2, Stelle 7

(A:A-9) = Feld 7, Stelle 7-9

(A:1-A) = Feld 7, Stelle 1-7

Restriktionen:

- Wird mit einer alphanumerischen Variablen indiziert, benutzt das System nur die rechten 4 Bit des EBCDIC-Codes.
- Der Inhalt der indizierenden Variablen darf nicht negativ oder größer 999 für ein Feld oder größer 99 für ein Teilfeld sein. Anderenfalls wird das Editor-Programm abgebrochen.

Die folgenden Programmbeispiele sollen die Verringerung des Programmieraufwands durch Anwendung der Indizierung verdeutlichen.

Beispiel 1:

Alle 15 Felder eines Satzes sollen auf dem Arbeitsplatzdrucker ausgegeben werden.

```
DECLARE␣ZZ.
MOVE␣1␣TO␣ZZ.
!A100␣TYPE␣(ZZ).
ADD␣1␣TO␣ZZ.
IF␣ZZ␣<␣15
   GOTO␣!A100.
RELEASE.
```

	Befehlsstruktur

Beispiel 2:

In einem 20stelligen Feld sollen alle Leerzeichen durch Sternchen (*) ersetzt werden.

```
DECLARE␣ZZ.
MOVE␣1␣TO␣ZZ.
!A100␣IF␣(1:ZZ)␣=␣' '
     MOVE␣'*'␣TO␣(1:ZZ).
ADD␣1␣TO␣ZZ.
IF␣ZZ␣<␣20
     GOTO␣!A100.
RELEASE.
```

5.2.4 Variable

Eine Variable ist ein Speicherbereich, der im Anwenderprogramm durch den DECLARE-Befehl definiert wird. Während des Programmablaufs stehen die Variablen als Zwischenspeicher für numerische Werte oder Zeichenketten zur Verfügung. Der Inhalt von Variablen kann in Datensätze übertragen oder ausgegeben werden.

Variable werden symbolisch durch Angabe ihres Namens adressiert. Der Name kann aus bis zu 8 Zeichen bestehen, wobei das erste Zeichen ein Buchstabe sein muß. Die restlichen Zeichen können Buchstaben (A-Z) oder Ziffern (0-9) sein.

Eine Variable kann numerische oder alphanumerische Werte aufnehmen.

- Numerische Variable sind max. 14 Stellen lang; das Vorzeichen wird auf der letzten Stelle verschlüsselt.
- Alphanumerische Variable sind max. 20 Stellen lang.

Durch den DECLARE-Befehl wird eine Variable 14stellig numerisch definiert und mit Nullen gefüllt. Während des Programmablaufs können Typ und Länge der Variablen verändert werden, wenn die Variable als Zieloperand eines MOVE-Befehls auftritt. In diesem Fall werden Typ und Länge des Zieloperanden durch Typ und Länge des Quelloperanden bestimmt. Die folgende Tabelle gibt Aufschluß über diese Zusammenhänge.

	Befehlsstruktur

Quell-Operand	Länge	Typ	Zieloperand Länge	Zieloperand Typ
Feld	1-99	AN	bis 20	AN
Teilfeld	1-99	AN	bis 20	AN
Numerisches Literal	14	N	14	N
Alphanum. Literal	1-120	AN	bis 20	AN
Variable	1-20	AN/N	wie Quell-Operand	wie Quell-Operand
Teilvariable	1-20	AN/N	wie Quell-Operand	wie Quell-Operand
Arithmetischer Ausdruck	14	N	14	N

Abhängig vom Programmtyp kann eine unterschiedliche Anzahl von Variablen benutzt werden:

bis 11 Variable im Feldende-Editor

bis 99 Variable im Stapelende-Editor, Ausgabe- und Sortierprogramm.

5.2.5 Teilvariable

Durch die Adressierung von Teilvariablen können einzelne Zeichen oder Zeichenfolgen innerhalb einer Variablen verarbeitet werden.

Eine Teilvariable wird folgendermaßen adressiert:

x:s-e

x = Name der Variablen

s = Startposition der Teilvariablen innerhalb der Variablen

e = Endeposition der Teilvariablen innerhalb der Variablen

Die Positionsangaben werden von links nach rechts gezählt. Wird nur ein Zeichen adressiert, ist der Adressausdruck x:s ausreichend.

Beispiel:

Inhalt der Variablen ORT 

Adressierung der Postleitzahl

ORT:1-4

Adressierung der letzten Stelle der Postleitzahl

ORT:4

Die Elemente s und e des Adressausdrucks können durch Variable dargestellt werden, dadurch ist eine Indizierung von Teilvariablen möglich (siehe auch indizierte Felder; Punkt 5.2.3).

Ist eine Teilvariable Zieloperand eines MOVE-Befehls, wird im Gegensatz zur Variablen Typ und Länge nicht verändert.

Befehlsstruktur

In folgenden Befehlen ist die Teilvariable als Operand zulässig:

- ADD (nicht erlaubt als Zieloperand)
- AUDIT
- DIVIDE (nicht erlaubt als Zieloperand)
- MOVE
- MULTIPLY (nicht erlaubt als Zieloperand)
- OUTPUT
- PAUSE
- SHOW
- SORT
- SUBTRACT (nicht erlaubt als Zieloperand)
- TYPE

Achtung:

Ist die Teilvariable Bestandteil eines arithmetischen Ausdrucks, so ist zu beachten, daß das System ein Minuszeichen, das bündig hinter der Startposition der Teilvariablen codiert wird, als "bis" interpretiert.

```
MOVE A:B-10+D TO C
bedeutet
Move Variable A, Stelle B bis 10 plus D nach C
und nicht
Move Variable A, Stelle B minus 10 plus D und C
```

Um alle Schwierigkeiten zu vermeiden, sollten alle Operanden eines arithmetischen Ausdrucks durch Leerzeichen getrennt werden.

5.2.6 Numerisches Literal

Anstelle einer symbolischen Speicheradresse (Feld, Variable) kann eine numerische Zeichenfolge unmittelbar als Operand eingesetzt werden.

Das numerische Literal ist max. 14 Stellen lang und darf aus den Ziffern 0-9 bestehen.

Handelt es sich um einen negativen Wert, wird die Ziffernfolge mit einer Oversign versehen. Das Oversign kann an jeder Stelle innerhalb der Ziffernfolge auftreten, außer an der 1 Stelle (das Literal würde sonst als Name einer Variablen interpretiert).

Beispiel:

```
ADD ␣ 4171 ␣TO␣ A.

MOVE ␣ 35689Z ␣TO␣ B.
```

5.2.7 Alphanumerisches Literal

Das alphanumerische Literal ist eine Zeichenfolge, die anstelle einer symbolischen Speicheradresse unmittelbar als Operand eingesetzt werden kann.

Das alphanumerische Literal ist max. 120 Stellen lang und wird entweder in Hochkomma (') oder Anführungsstriche (") eingeschlossen. Innerhalb des Literals kann jedes beliebige Zeichen auftreten, ausgenommen die Zeichen, in die das Literal eingeschlossen ist.

Beispiel:

```
PAUSE␣'WENN ENDE,␣"Y"␣EINGEBEN'.

oder

PAUSE␣"WENN␣ENDE,␣'Y'␣EINGEBEN".
```

Ein Literal, das aus mehreren identischen Zeichen besteht, kann in abgekürzter Schreibweise codiert werden. Zu diesem Zweck wird die Länge des Literals als Zahl und sofort anschließend das Zeichen selbst - in Hochkomma oder Anführungszeichen eingeschlossen - angegeben.

Beispiel:

```
OUTPUT␣120'-'.
```

oder

```
OUTPUT␣20"'".
```

5.2.8 Arithmetischer Ausdruck

Ein arithmetischer Ausdruck setzt sich aus zwei oder mehreren Operanden wie Feld, Variable oder Literal zusammen. Die einzelnen Operanden werden durch arithmetische Operatoren miteinander verknüpft.

Die möglichen Operatoren sind:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division

In einem arithmetischen Ausdruck können beliebig viele Operatoren enthalten sein.

Der Ausdruck wird grundsätzlich von links nach rechts interpretiert. Die Hierarchie der mathematischen Operationen wird nicht berücksichtigt.

Wird das Ergebnis eines arithmetischen Ausdrucks in einer Variablen abgestellt, hat die Variable immer eine Länge von 14 Stellen. Der Wert wird rechtsbündig abgestellt und die Variable mit Nullen aufgefüllt.

Befehlsstruktur

Wird das Ergebnis in einem Feld abgestellt, bleibt die Länge des Feldes unverändert.

Die einzelnen Operanden und Operatoren des arithmetischen Ausdrucks sollten durch Leerzeichen getrennt werden.

Beispiele:

```
MOVE␣(1)␣+␣(2)␣TO␣A.
```
Addition der Felder (1) und (2). Ergebnis in der Variablen A.

```
MOVE␣MENGE␣*␣PREIS␣*␣RABATT␣/␣100␣TO␣WERT.
```
Multiplikation der Variablen MENGE, PREIS, RABATT und Division durch 100. Ergebnis in der Variablen WERT.

```
MOVE␣(21)␣/␣90␣+␣TEMP␣-␣(6)␣TO␣A.
```
Division des Feldes (21) durch 90. Zum Ergebnis wird die Variable TEMP addiert und davon Feld (6) subtrahiert. Das Ergebnis wird in der Variablen A abgestellt.

Wenn möglich, sollten arithmetische Ausdrücke anstelle arithmetischer Operationen benutzt werden, da dadurch der Speicherbedarf der Programme geringer wird.

5.2.9 Befehlsergänzung

Die Output-Ergänzungen (siehe dort) <JOB>, <FILE>, <DATE>, <TIME>, <KEY> und <FMT> können in allen Befehlen, in denen ein Alpa-Literal als Operand zulässig ist, benutzt werden.

Z.B. MOVE, IF, SHOW etc.

Hiermit ist die Möglichkeit gegeben, das Systemdatum, den Stapelnamen, den Standard-Job-Namen, die Nummer des Eingabeformats, den Indexschlüssel und die Uhrzeit in ein Feld oder eine Variable zu bringen und damit weiter zu arbeiten.

5.3 Arithmetische Operationen

Im System können die arithmetischen Operationen

- Addieren (ADD)
- Subtrahieren (SUBTRACT)
- Multiplizieren (MULTIPLY)
- Dividieren (DIVIDE)

vorzeichengerecht durchgeführt werden. Das Vorzeichen wird bei numerischen Werten auf der letzten Stelle mit verschlüsselt (positives Vorzeichen = F, negatives Vorzeichen = D).

	Befehlsstruktur

Bei den Operanden unterscheidet man zwischen Quelloperand und Zieloperand. Der Quelloperand bleibt unverändert, im Zieloperand steht das Ergebnis.

Der Quelloperand kann aus folgenden Ausdrücken bestehen:

- numerisches, vom Eingabeformat definiertes Feld
- numerisches Literal
- arithmetischer Ausdruck
- Variable

Zieloperand kann nur eine Variable sein.

Beispiel:

```
DIVIDE␣100␣INTO␣RABATT.
MULTIPLY␣MENGE␣TIMES␣PREIS.
```

Es werden nur ganzzahlige Werte verarbeitet. Verschiebungen der Dezimalwertigkeiten können durch Multiplikation und Division mit 10er-Potenzen erreicht werden.

Ist das Ergebnis der arithmetischen Operation größer als 14 Stellen, findet ein Überlauf statt und es wird ein Überlauf-Merker gesetzt, der vom Programm abgefragt werden kann (WHEN OVERFLOW...).

Mit dem Teilergebnis sollte nicht weitergerechnet werden. Auf dem Bildschirm erscheint in der Fehlerzeile die Meldung ARITHMETISCHER ÜBERLAUF und der Fehlerton ertönt.

Achtung:

Die Angabe eines Feldes als Zieloperand bei arithmetischen Operationen ist nicht möglich.

Sollen Ergebnisse einer arithmetischen Operation in einem Feld abgestellt werden, so kann ein Transportbefehl mit einem arithmetischen Ausdruck benutzt werden.

```
Beispiel:   ADD␣(1)␣TO␣(2). = nicht möglich
            MOVE␣(1)␣+␣(2)␣TO␣(2). = möglich
```

	Befehlsstruktur

5.4 Befehlsformat

Zur Beschreibung des Befehlsformats gelten folgende Regeln:

- Der Befehl wird gemäß seiner Formatbeschreibung von links nach rechts codiert.
- Zwingend notwendige Angaben sind in Großbuchstaben geschrieben.
- Kleingeschriebene Wörter kennzeichnen Begriffe, die an dieser Stelle einzusetzen sind.
- Angaben in eckigen Klammern können - falls sie nicht benötigt werden - entfallen.
- Stehen mehrere Angaben in geschweiften Klammern übereinander, muß eine dieser Angaben codiert werden.

5.5 Codierung

Die Befehle eines Programms werden auf dem Formular "EDITOR-PROGRAMM" codiert. Jede Zeile des Formulars ist 40 Stellen lang. Belegt ein Befehl mehr als eine Zeile des Formulars, kann er in der Folgezeile fortgesetzt werden. Dabei können auch Operatoren oder Operanden getrennt werden.

Beispiel:

```
---------- MOVE␣SUMME␣TO␣ENDS
UMME.

oder

---------- MOVE␣SUMME␣TO␣␣␣␣
ENDSUMME.
```

Jeder Befehl muß mit einem Punkt oder einem Semikolon abgeschlossen werden. Das Semikolon dient zur Trennung logisch zusammengehörender Befehle nach einem IF, WHEN oder ELSE-Befehl.

	Befehlsstruktur

Beispiel:

```
IF␣ A␣>␣B

    MOVE␣A␣TO␣B;

    SUBTRACT␣A␣FROM␣C.

MOVE␣B␣TO␣C.
```

Ist die Bedingung A > B erfüllt, werden die folgenden Befehle abgearbeitet. Ist die Bedingung nicht erfüllt, wird der Programmablauf mit dem auf den Punkt folgenden Befehl fortgesetzt.

Nach einem Punkt können beliebig viele Leerzeichen folgen.

Die einzelnen Elemente eines Befehls können durch Komma und Leerzeichen oder nur durch Leerzeichen getrennt werden. Zwischen den Elementen können beliebig viele Leerzeichen stehen.

5.6 Befehle

5.6.1 ACCEPT

Format:

```
ACCEPT␣[<LOC␣Zeile[,Spalte]>]␣Variable␣
                [ELSE␣Befehl[;␣Befehl]...].
```

Funktion:

Der ACCEPT-Befehl stoppt den Programmablauf und erlaubt die Eingabe von Daten in eine Variable. Die Dateneingabe kann an beliebiger Stelle des Bildschirms erfolgen.

- <LOC Zeile [, Spalte]> definiert die Position der Eingabe auf dem Bildschirm. Als Eingabezeile kann - abhängig vom angeschlossenen Bildschirm - Zeile 1-24 codiert werden. Zeile 1 ist in diesem Fall die Statuszeile, Zeile 2 die Fehlerzeile. Beide Zeilen werden ggfs. überschrieben (Vorsicht!). Als Spalte kann - abhängig vom angeschlossenen Bildschirm - Spalte 1-80 codiert werden. Wird Spalte nicht codiert, beginnt die Eingabe auf Spalte 1. Entfällt die Angabe LOC, beginnt die Eingabe in Zeile 2, Spalte 1 (Fehlerzeile).

- Variable

  Abhängig vom Typ der Variablen erlaubt das System numerische oder alphanumerische Eingaben. Die Anzahl einzugebender Zeichen wird durch die Länge der Variablen bestimmt.

  Durch einen MOVE-Befehl können Typ und Länge der Variablen vor Ausführung des ACCEPT-Befehls festgelegt werden.

  Soll eine numerische Variable kürzer als 14 Stellen definiert werden, bietet sich folgender Kunstgriff an:

```
MOVE '␣'  TO  A.
ADD  0    TO  A.
```

  Das Resultat dieser Operation ist eine 1-stellige, numerische Variable.

	Befehlsstruktur	ACCEPT

[ELSE Befehl [; Befehl]...]

Der ELSE-Zweig des Befehls wird ausgeführt, wenn die DIAL-Taste betätigt wird. Ist der ELSE-Zweig nicht codiert, zeigt das System bei Betätigung der DIAL-Taste einen Fehler an.

Beispiel:

```
DECLARE␣DATUM.
MOVE␣'␣␣␣␣␣␣␣'␣TO  DATUM.
ADD␣0␣TO␣DATUM.
SHOW␣ <LOC␣3, 1>␣' DATUM:'.
ACCEPT␣ <LOC␣3,8>␣ DATUM
        ELSE␣MOVE␣10177␣TO␣DATUM.
```

5.6.2 ADD

Format:

ADD ␣ { Literal / Feld / Variable / Arithmetischer Ausdruck } ␣TO␣Variable.

Funktion:

Beide Operanden werden addiert. Das Ergebnis wird rechtsbündig im zweiten Operanden abgestellt. Der erste Operand bleibt unverändert.

Die Länge des zweiten Operanden bleibt erhalten. Ist das Ergebnis länger als der zweite Operand, findet ein Oberlauf statt und der Oberlauf-Merker wird gesetzt. Der Merker kann vom Programm abgefragt werden. Der Oberlauf geht verloren.

Wird ein alphanumerischer Operand verwendet, rechnet das System mit den rechten 4 Bit des EBCDIC-Codes.

Beispiel:

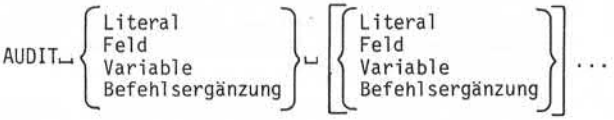
TOTAL = |0|0|1|8|4|4|
(2) = |0|0|0|0|2|4|
@3@ = |0|0|1|5|9|

- ADD ␣ @3@ ␣ TO ␣TOTAL.
  TOTAL = |0|0|2|0|0|3|

- ADD␣ (2) ␣ * ␣@3@ ␣ TO ␣TOTAL.
  TOTAL = |0|0|5|6|6|0|

5.6.3 AUDIT

Format:



Funktion:

Der AUDIT-Befehl gibt den aktuellen Datensatz auf Magnetband aus (z.B. als Datensicherung).

- Der AUDIT-Befehl kann im Feldende-/Stapelende-Editor, im Ausgabeprogramm und im Sortierprogramm verwendet werden.
- Die Ausgabe erfolgt grundsätzlich auf Magnetbandeinheit 4 (D). Die Zuweisung muß entsprechend festgelegt werden.
- Magnetbandstation 4 ist während des Programmlaufs mit AUDIT-Funktion für alle anderen Operationen gesperrt.
- Es kann nur eine 9-Kanal-Magnetbandeinheit verwendet werden.
- Mehrere Bildschirmarbeitsplätze können gleichzeitig AUDIT-Funktion ausführen. Der Programmierer muß daher Programme erstellen, welche die auf dem AUDIT-Band gesammelten Daten wieder trennen.
- Bei Anwendung des AUDIT-Befehls im Feldende-Editor werden im Prüfmodus nur Sätze ausgegeben, in denen Korrekturen vorgenommen wurden.
- Die Überprüf- und ABL-Funktion ignoriert den AUDIT-Befehl.
- Die maximale Länge eines AUDIT-Satzes beträgt 200 Stellen.
- Durch die Formatnummern wird beim Wiedereinlesen das Eingabeformat bestimmt. Aus diesem Grund muß darauf geachtet werden, daß Formate mit gleicher Nummer auch mit gleicher Satzlänge ausgegeben werden.
- Die möglichen Befehlsergänzungen werden im Abschnitt OUTPUT-Befehlsergänzungen (5.6.26) erläutert.
- Im Überprüf-Modus und in der ABL-Funktion wird der AUDIT-Befehl ignoriert.

Beispiel:

Daten werden mit zwei Standard-Jobs erfaßt. JOB1 enthält zwei Eingabeformate mit 20 und 30 Zeichen Länge; die Stapelnamenvorgabe ist ABC*******. JOB2 enthält ebenfalls zwei Eingabeformate mit 40 und 50 Zeichen Länge; die Stapelnamenvorgabe ist XYZ*******.

Durch das folgende Feldende-Editorprogramm werden die Daten während der Erfassung auf Band gesichert:

```
AUDIT␣ <FMT> ␣ <BATCH> ␣ <ALL> ␣ <SKIP␣61> .
RELEASE.
```

Für das Einlesen des Sicherungsbandes existiert ein Standard-Job mit einem Eingabeformat, das aus 3 Feldern besteht.

- Feld 1 = Formatnummer, 1-stellig numerisch
- Feld 2 = Stapelname, 10-stellig alphanumerisch
- Feld 3 = Daten, 50-stellig alphanumerisch

Die eingelesenen Sicherungsdaten werden mit dem folgenden Sortierprogramm nach Stapelnamen und Formatnummern aufsteigend sortiert:

```
SORT␣ (2)␣ (1).
RELEASE.
```

Der sortierte Stapel wird anschließend mit RESCUE-Kennung auf Band ausgegeben.

```
DECLARE␣NAME.
IF␣ (2)␣ = ␣NAME
       GOTO  !A100.
OUTPUT ␣'!'␣<DEFER> .
IF␣ (2:1-3)␣=␣'ABC'
       OUTPUT␣ 'JOB1␣␣␣␣'␣(2).
IF␣ (2:1-3)␣= ␣ ' XYZ'
       OUTPUT␣ 'JOB2␣␣␣␣'␣(2).
MOVE␣ (2)␣ TO␣NAME.
!A100
IF␣ (2:1-3)␣= ␣ 'XYZ'
       GOTO␣!A200.
```

```
IF␣(1)␣=␣1
       OUTPUT␣(1)␣(3:1-20).
IF␣(1)␣=␣2
       OUTPUT␣(1)␣(3:1-30).
RELEASE.
!A200
IF␣(1)␣=␣1
       OUTPUT␣(1)␣(3:1-40).
IF␣(1)␣=␣2
       OUTPUT␣(1)␣(3:1-50).
RELEASE.
```

Anschließend wird das Band mit RESCUE-Kennung gelesen.

Damit ist der Zustand vor dem Systemzusammenbruch wieder hergestellt.

	Befehlsstruktur	BACK (Stapel)

5.6.4.1 BACK (Stapel)

Format:

BACK␣[ELSE␣Befehl[;␣Befehl]...].

Wirkung:

Der Back-Befehl blättert im Stapel um 1 Satz zurück und erlaubt den Zugriff auf den vorhergehenden Satz. Mehrfaches Zurückblättern ist möglich. Wird versucht über den 1.Satz des Stapels zurückzublättern, führt das System den ELSE-Zweig des Befehls aus. Ist der ELSE-Zweig nicht codiert, erscheint eine Fehlermeldung und das Editor-Programm wird abgebrochen.

Jede Adressierung von Feldern durch das Programm bezieht sich auf den Satz, auf den zurückgeblättert wurde (mit zugehörigem Eingabeformat).

Im Feldende-Editor bleibt die Bildschirmanzeige bei Ausführung des BACK-Befehls unverändert, da der Satz, auf den zurückgeblättert wurde, nicht angezeigt wird.

Nach Ausführung einer RELEASE-Instruktion im Feldende-Editor ist der aktuelle Satz wieder im Zugriff.

In allen anderen Programmen wird auf den folgenden Satz zugegriffen.

Wird im Feldende-Editor vor den RELEASE-Befehl ein POSITION-Befehl durchgeführt, wird der Satz, auf den zurückgesetzt wurde, zum aktuellen Satz.

Beispiel:

In einem Feldende-Editor soll der Inhalt von Feld 3 des aktuellen Satzes in Feld 3 des vorhergehenden Satzes übertragen werden.

```
DECLARE␣A.
MOVE␣(3)␣TO␣A.
BACK␣ELSE␣RELEASE.
MOVE␣A␣TO␣(3).
RELEASE.
```

5.6.4.2 BACK (Indexdatei)

Format:

BACK ␣Dateiname␣[ELSE ␣Befehl[; ␣Befehl]...].

Funktion:

Dieser Befehl greift im Datenstapel einer Indexdatei auf den physikalisch vorhergehenden Satz zu. Dadurch wird nur der Satzzeiger verändert. Der Indexzeiger bleibt unbeeinflußt. Durch einen anschließenden GET CURRENT - Befehl kann daher wieder auf den letzten über den Index gelesenen Satz zugegriffen werden (siehe auch Abschnitt 4.1.2 und 4.2).

Der ELSE-Zweig des Befehls wird ausgeführt, wenn versucht wird, über den ersten Satz des Datenstapels hinauszulesen. Ist der ELSE-Zweig nicht codiert, erscheint in diesem Fall eine Fehlermeldung und das Programm wird abgebrochen.

Achtung! In Indexdateien, die aus mehreren Datenstapeln zusammengesetzt sind (siehe Abschnitt 4.1.3), kann durch den BACK-Befehl nicht nacheinander auf alle Sätze der Datei zugegriffen werden. Der Zugriff ist durch die Grenzen der einzelnen Stapel limitiert. Um auf alle Sätze der Datei zugreifen zu können, muß eine Kombination aus GET und BACK angewendet werden.

5.6.5 BYPASS

Format:

BYPASS ␣ [ELSE ␣ Befehl [; ␣ Befehl]...].

Funktion:

Der BYPASS-Befehl beendet die Verarbeitung des aktuellen Stapels. Danach wird mit der Verarbeitung des nächsten Stapels - soweit vorhanden - begonnen. BYPASS wird im wesentlichen bei Multi-Batch-Verarbeitung verwendet. Im Anschluß an den BYPASS-Befehl wird das Anwenderprogramm vom Beginn an durchlaufen (Wirkung wie RELEASE). Der Unterschied zum RELEASE-Befehl besteht darin, daß BYPASS stapelbezogen, RELEASE dagegen satzbezogen arbeitet.

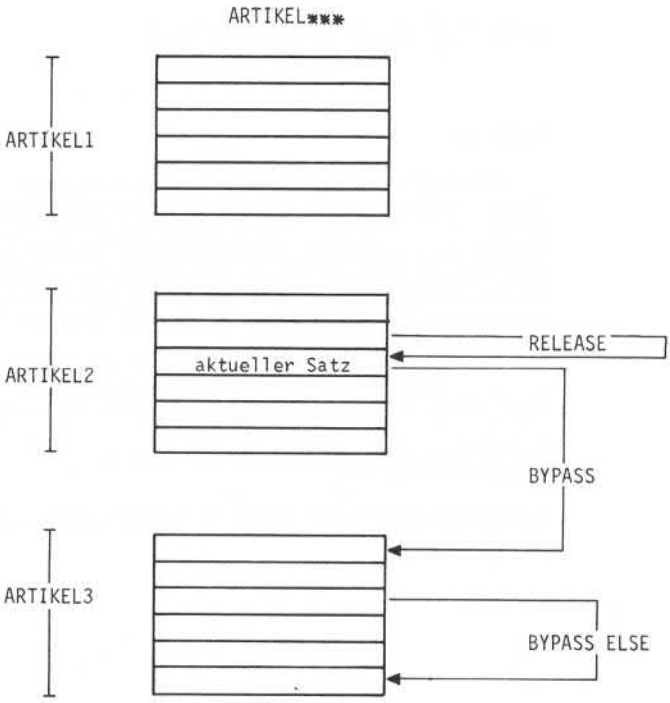
Der ELSE-Zweig wird durchlaufen, wenn der BYPASS-Befehl während der Verarbeitung des letzten Stapels ausgeführt wird. Ist der ELSE-Zweig nicht codiert, wird in diesem Fall das Programm beendet.

Nach Durchlaufen des ELSE-Zweiges ist der letzte Satz des letzten Stapels im Zugriff.

Der BYPASS-Befehl darf nicht im Feldende-Editor verwendet werden.

	Befehlsstruktur	BYPASS

Beispiel:



5.6.6 CLEAR

Format:

CLEAR␣Feld.

Funktion:

Der CLEAR-Befehl löscht ein Fehlerkennzeichen (#), das zuvor entweder vom Programm oder von der Bedienungskraft in das betreffende Feld gesetzt wurde.

Anstelle des Fehlerkennzeichens tritt ein ↻ (Blank).

Wird die Angabe Feld als Teilfeld mit 1 Zeichen Länge codiert, (z.B. CLEAR (3:4)), löscht das System ein auf dieser Zeichenposition gesetztes Fehlerkennzeichen.

Wird dagegen das gesamte Feld angegeben (z.B. CLEAR (3)), löscht das System ein Fehlerkennzeichen, das auf der von links gesehenen 1.Stelle des Feldes gesetzt ist.

Befindet sich auf der angegebenen Position kein Fehlerkennzeichen, bleibt das Feld unverändert.

Beispiel:

Feld 3 = |S|C|H|U|#|Z|E|

CLEAR␣(3:5).

Feld 3 = |S|C|H|U|␣|Z|E|

Feld 4 = |#|5|4|3|8|2|6|

CLEAR␣(4).

Feld 4 = |␣|5|4|3|8|2|6|

5.6.7 CLOSE

Format:

CLOSE␣Dateiname.

Funktion:

Der CLOSE-Befehl schließt eine eröffnete Indexdatei ab. CLOSE muß nur in Programmen angewendet werden, die mehr als 4 Indexdateien verarbeiten. Vor Eröffnung der 5.Datei muß eine der 4 bereits eröffneten Dateien mit CLOSE abgeschlossen werden.

Beispiel:

```
OPEN DATEI-A.
OPEN DATEI-B.
OPEN DATEI-C.
OPEN DATEI-D.
      \
       Verarbeitung
      /
CLOSE DATEI-B.
OPEN DATEI-E.
```

5.6.8 DECLARE

Format:

DECLARE␣Variable␣[Variable]...

Funktion:

Durch den DECLARE-Befehl werden die Namen der im Programm benutzten Variablen festgelegt. Der DECLARE-Befehl muß dem ersten Ansprechen der Variablen vorausgehen.

Der Name einer Variablen besteht aus max. 8 Zeichen, wobei Buchstaben und Ziffern verwendet werden können. Das erste Zeichen des Namens muß ein Buchstabe sein.

Durch den DECLARE-Befehl wird eine Variable als 14-stelliges numerisches Feld initialisiert und mit Nullen gefüllt. Länge und Typ der Variablen können durch einen MOVE-Befehl verändert werden (siehe Abschnitt 5.2.4).

Der DECLARE-Befehl wird nur beim ersten Programmdurchlauf wirksam. In allen folgenden Programmdurchläufen werden die deklarierten Variablen weder gelöscht noch verändert.

Alphanumerische Variable sind max. 20 Stellen, numerische Variable max. 14 Stellen lang.

Die maximale Anzahl von Variablen beträgt 11 im Feldende-Editor und 99 in allen anderen Programmen.

Achtung! Treten bei sehr großen Programmen Zeitprobleme auf, sollte folgendes berücksichtigt werden:

- Nach dem Ansprechen von jeweils 11 Variablen wird die Verarbeitungsgeschwindigkeit vermindert, da 11 Variable mit einem Plattenzugriff gelesen werden. Bei der Verarbeitung der ersten 11 Variablen bleibt die Verarbeitungsgeschwindigkeit also konstant; wird die 12.Variable angesprochen, sinkt die Geschwindigkeit des Systems etwas. Bei Variable 12-22 bleibt die verminderte Geschwindigkeit konstant. Ein weitere Verlangsamung erfolgt beim Ansprechen ab Variable 23 usw.

	Befehlsstruktur	DECLARE

- Um eine schnellere Abarbeitung der Programme zu erreichen, sollten die Variablen in der Reihenfolge ihrer Abarbeitung innerhalb der Programme definiert werden.

Beispiel:

Im Programm werden Variable in der Folge B, M, X, A, S, K benötigt. Die Definition der Variablen sollte lauten:

DECLARE␣B␣M␣X␣A␣S␣K.

und nicht

DECLARE␣A␣B␣K␣M␣S␣X.

5.6.9 DEFINE

Format:

DEFINE␣Dateiname␣ [Dateiname]...

Funktion:

Durch diesen Befehl werden die in einem Programm verwendeten Indexdateien definiert. Eine Datei, die nicht definiert wurde, kann nicht eröffnet werden.

Es ist möglich, bis zu 64 Dateien in einem Programm zu definieren, jedoch können nur 4 Dateien gleichzeitig eröffnet sein.

5.6.10 DELETE

Format:

DELETE␣Dateiname.

Funktion:

Der DELETE-Befehl entfernt den durch den Satzzeiger adressierten Satz aus dem Datenstapel einer Indexdatei. Der Satzzeiger bleibt unverändert. Der gelöschte Satz kann anschließend weder durch GET, noch durch FORWARD- oder BACK-Befehle gelesen werden. Bestehende Verkettungen des Satzes mit Indizes werden automatisch durch das Betriebssystem gelöscht.

Achtung!
- Vor Ausführung des DELETE-Befehls muß sichergestellt werden, daß der Satzzeiger auf den richtigen Satz verweist.
- Der mit dem Datenstapel verbundene Index wird durch DELETE nicht verändert.

Beispiel:

```
GET␣ARTIKEL␣USING␣ARTNR
        ELSE␣GOTO␣ !FEHLER.
DELETE␣ARTIKEL.
```

5.6.11 DIVIDE

Format:

DIVIDE ␣ { Literal / Feld / Variable / Arithmetischer Ausdruck } ␣ INTO␣Variable.

Funktion:

Der Inhalt des zweiten Operanden wird durch den Inhalt des ersten Operanden dividiert. Das Ergebnis wird im zweiten Operanden abgestellt. Der erste Operand bleibt unverändert. Die Länge des zweiten Operanden bleibt erhalten.

Der Rest der Division geht ohne Meldung verloren.

Eine Division durch Null ergibt Null.

Wird ein alphanumerischer Operand verwendet, rechnet das System mit den rechten 4 Bit des EBCDIC-Codes.

Beispiel:

PREIS = |0|3|8|1|6|

(2) = |0|0|0|2|4|

DIVIDE␣(2)␣INTO␣PREIS.

PREIS = |0|0|1|5|9|

5.6.12 FLAG

Format:

FLAG␣Feld.

Funktion:

Der FLAG-Befehl setzt in einem vom Eingabeformat definierten Feld ein Fehlerkennzeichen (#).

Wird nur die Feldnummer - z.B. FLAG (2) - angegeben, wird das Fehlerkennzeichen auf der ersten Stelle des Feldes (links) gesetzt.

Darüberhinaus kann das Fehlerkennzeichen auf einer bestimmten Stelle innerhalb des Feldes gesetzt werden.

Beispiel:

FLAG␣(2).

Im Feld 2 wird auf Stelle 1 ein Fehlerkennzeichen gesetzt.

FLAG␣(2:3).

Im Feld 2 wird auf Stelle 3 ein Fehlerkennzeichen gesetzt.

5.6.13.1 FORWARD (Stapel)

Format:

FORWARD␣[ELSE␣Befehl[;␣Befehl]...].

Funktion:

Der FORWARD-Befehl blättert im Stapel um 1 Satz vorwärts und erlaubt den Zugriff auf den nächsten Satz. Mehrfaches Vorwärtsblättern ist möglich.

Wird versucht,über den letzten Satz des Stapels hinauszublättern, führt das System den ELSE-Zweig des Befehls aus. Ist der ELSE-Zweig nicht codiert, erscheint in diesem Fall eine Fehlermeldung und das Programm wird abgebrochen.

Jede Adressierung von Feldern durch das Programm bezieht sich auf den Satz, auf den vorwärtsgeblättert wurde (mit zugehörigem Eingabeformat).

Im Feldende-Editor bleibt die Bildschirmanzeige bei Ausführung des FORWARD-Befehls unverändert, da der Satz, auf den vorwärtsgeblättert wurde, nicht angezeigt wird.

Nach Ausführung einer RELEASE-Instruktion im Feldende-Editor ist der aktuelle Satz wieder im Zugriff.

In allen anderen Programmen wird auf den folgenden Satz zugegriffen.

Wird im Feldende-Editor vor dem RELEASE-Befehl ein POSITION-Befehl ausgeführt, wird der Satz, auf den vorgeblättert wurde, zum aktuellen Satz.

Im Feldende-Editor ist die Anwendung des FORWARD-Befehls im Eingabe-Modus nicht sinnvoll, da kein folgender Satz existiert, auf den zugegriffen werden kann.

	Befehlsstruktur	FORWARD (Stapel)

Beispiel:

In einem Feldende-Editor soll der Inhalt von Feld 3 des aktuellen Satzes in Feld 3 des folgenden Satzes übertragen werden.

```
DECLARE␣A.
MOVE␣(3)␣TO␣A.
FORWARD␣ELSE␣RELEASE.
MOVE␣A␣TO␣(3).
RELEASE.
```

5.6.13.2 FORWARD (Indexdatei)

Format:

FORWARD␣Dateiname␣[ELSE␣Befehl[;␣Befehl]...].

Funktion:

Dieser Befehl greift im Datenstapel einer Indexdatei auf den nachfolgenden Satz zu. Dadurch wird nur der Satzzeiger verändert. Der Indexzeiger bleibt unbeeinflußt. Durch einen anschliessenden GET CURRENT - Befehl kann daher wieder auf den letzten über den Index gelesenen Satz zugegriffen werden (siehe auch Abschnitt 4.1.2 und 4.2).

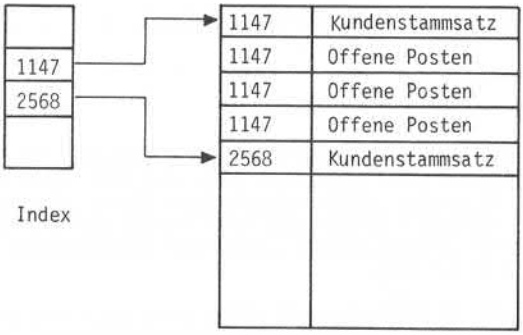
Der ELSE-Zweig des Befehls wird ausgeführt, wenn versucht wird, über den letzten Satz des Datenstapels hinauszulesen. Ist der ELSE-Zweig nicht codiert, erscheint in diesem Fall eine Fehlermeldung und das Programm wird abgebrochen.

Achtung! In Indexdateien, die aus mehreren Datenstapeln zusammengesetzt sind (siehe Abschnitt 4.1.3), kann durch den FORWARD-Befehl nicht nacheinander auf alle Sätze der Datei zugegriffen werden. Der Zugriff ist durch die Grenzen der einzelnen Stapel limitiert. Um auf alle Sätze der Datei zugreifen zu können, muß eine Kombination aus GET und FORWARD angewendet werden.

	Befehlsstruktur	FORWARD (Indexdatei)

Beispiel:

Eine Indexdatei hat folgenden Aufbau:



Alle zu einem Kunden gehörenden Offenen Posten sollen gelesen werden:

```
DECLARE␣NUMMER␣SUMME.
DEFINE␣KUSTAMM.
OPEN␣KUSTAMM.
MOVE␣'␣␣␣␣'␣TO␣NUMMER.
!A100
ACCEPT␣NUMMER.
GET␣KUSTAMM␣USING␣NUMMER
          ELSE␣GOTO␣!A100.
```

	Befehlsstruktur	FORWARD (Indexdatei)

```
!A200
FORWARD␣KUSTAMM
        ELSE␣GOTO␣!ENDKUND.
IF␣@1@␣≠␣NUMMER
        GOTO␣!ENDKUND.
ADD␣@2@␣TO␣SUMME.
GOTO␣!A200.
!ENDKUND
SHOW␣SUMME|LZ.
MOVE␣0␣TO␣SUMME.
GOTO␣!A100.
```

5.6.14 GET

Format:

GET␣Dateiname␣USING␣{Literal / Variable / Feld}␣ELSE␣Befehl[;␣Befehl]...

Funktion:

Durch den Befehl GET wird der Satz einer Datei gelesen, dessen Schlüssel im Index mit dem nach USING angegebenen Suchargument übereinstimmt. Besitzen zwei oder mehrere Sätze den gleichen Schlüssel, wird immer der erste Satz gelesen. Durch die Instruktion GET NEXT können die folgenden Sätze gelesen werden.

Das Suchargument darf keine Teilvariable und kein Teilfeld sein. Sollen Teilvariable/Teilfelder oder ein aus mehreren Operanden zusammengesetztes Suchargument verwendet werden, müssen diese in eine Variable übertragen und anschließend die Variable als Suchargument benutzt werden.

Ist das Suchargument kürzer als der Schlüssel im Index, ist bei EBCDIC-Schlüsseln Gleichheit gegeben, wenn beide Begriffe bis zum Ende des Suchargumentes übereinstimmen. Der umgekehrte Fall ist nicht möglich.

Unter der Voraussetzung, daß der Index im EBCDIC-Code aufgebaut ist, kann im Suchargument mit der Sternchen-Vereinbarung gearbeitet werden.

Andere, spezialisierte Formen des GET-Befehls ermöglichen den indexsequentiellen Zugriff auf eine Datei:

- GET␣FIRST␣Dateiname␣ELSE␣Befehl [;␣Befehl]...

  Liest den logisch ersten, im Index enthaltenen Satz.

- GET␣LAST␣Dateiname␣ELSE␣Befehl [;␣Befehl]...

  Liest den logisch letzten, im Index enthaltenen Satz.

	Befehlsstruktur	GET

- GET␣NEXT␣ Dateiname␣ELSE␣Befehl [;␣Befehl]...

  Liest den logisch nächsten, im Index enthaltenen Satz.

- GET␣PRIOR␣ Dateiname␣ELSE␣Befehl [;␣Befehl]...

  Liest den logisch vorhergehenden, im Index enthaltenen Satz.

- GET␣CURRENT␣ Dateiname␣ELSE␣Befehl [;␣Befehl]...

  Liest den Satz, auf den der Indexzeiger verweist.

Der ELSE-Zweig muß zwingend in allen Varianten des GET-Befehls codiert werden. Die folgende Tabelle zeigt, wann der ELSE-Zweig durchlaufen wird:

GET /USING	Schlüssel nicht gefunden
GET FIRST	Index leer
GET LAST	Index leer
GET NEXT	Ende des Index überschritten
GET PRIOR	Ende des Index unterschritten
GET CURRENT	Schlüssel gelöscht

Achtung! Alle Varianten des GET-Befehls verändern den Indexzeiger und den Satzzeiger. Wird der ELSE-Zweig eines Befehls durchlaufen, sind beide Zeiger zerstört.

5.6.15 GOTO

Format:

GOTO␣Label.

Funktion:

Durch den Befehl GOTO wird der lineare Programmablauf verlassen und das Programm verzweigt mit einem Vorwärts- oder Rückwärtssprung zum angegebenen Label. Der Label kann aus 2 bis 9 Zeichen bestehen. Das erste Zeichen muß ein Ausrufungszeichen sein, danach folgt zwingend ein Buchstabe. Der Rest des Labels kann aus Buchstaben oder Ziffern bestehen. Die Anzahl der Label in einem Programm ist nicht begrenzt.

Der im GOTO-Befehl codierte Label wird an der Stelle des Programms wiederholt, auf die verzweigt werden soll.

Der Label kann vor oder hinter dem GOTO-Befehl liegen, d.h. sowohl Vorwärts- als auch Rückwärtssprünge sind möglich.

Ein Label, der im Programm als Ansprungpunkt codiert wird, darf nicht durch einen Punkt abgeschlossen werden.

Achtung! Der GOTO-Befehl ist nur für Sprünge innerhalb eines Hauptprogramms oder eines Unterprogramms zulässig. Für einen Wechsel zwischen Haupt- und Unterprogramm darf er nicht benutzt werden.

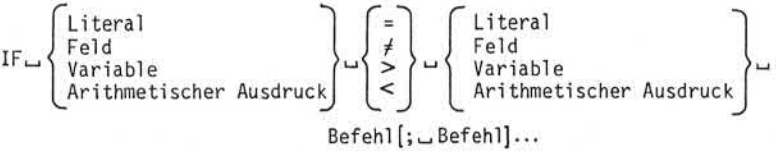
Beispiel:

```
DECLARE␣ZAEHLER␣SUMME.
!A100
ADD␣1␣TO␣ZAEHLER.
IF␣ZAEHLER␣>10␣GOTO␣!A200.
ADD␣(ZAEHLER)␣TO␣SUMME.
GOTO␣!A100.
!A200
SHOW␣SUMME.
STOP.
```

	Befehlsstruktur	IF

5.6.16 IF

Format:



Funktion:

Der Befehl IF vergleicht zwei Operanden und führt den folgenden Befehl aus, wenn die Vergleichsbedingung erfüllt ist.

Ist die Bedingung nicht erfüllt, werden alle Befehle bis zum nächsten Punkt übergangen und erst der auf den Punkt folgende Befehl ausgeführt.

In einem IF-Satz können mehrere, durch Semikolon (;) getrennte Befehle enthalten sein. Der letzte Befehl des IF-Satzes muß durch einen Punkt abgeschlossen sein. Innerhalb eines IF-Satzes dürfen keine weiteren Bedingungsabfragen wie IF, WHEN oder ELSE auftreten. Der Befehl EXIT ist nach dem Semikolon nicht erlaubt.
Die möglichen Vergleichsbedingungen sind:

=	gleich
≠	ungleich
>	größer
<	kleiner

	Befehlsstruktur	IF

Beispiel:

```
IF␣(1)␣>␣SUMME
        PAUSE␣'FEHLER';
        MOVE␣(1)␣-␣SUMME␣TO␣DIFF;
        GOTO␣!A100.
SHOW␣'KEINE␣DIFFERENZ'.
```

Wenn die Bedingung (1) > SUMME erfüllt ist, wird mit der Verarbeitung des PAUSE-Befehls fortgefahren; ist die Bedingung nicht erfüllt, werden alle Befehle bis zum Punkt übergangen und erst beim Befehl SHOW das Programm fortgesetzt.

Ein IF-Befehl kann zwei unterschiedliche Vergleiche durchführen:

- Ein alphanumerischer Vergleich wird ausgeführt, wenn beide Operanden alphanumerisch sind. In diesem Fall werden die Operanden von links nach rechts fortlaufend verglichen. Ist ein Operand kürzer als der andere, wird er rechts mit imaginären Leerzeichen aufgefüllt.

  Der Vergleich wird anhand des internen Maschinencodes durchgeführt. Die folgende Tabelle zeigt die Hierarchie des Maschinencodes:

-	Leerzeichen	-	S-Z
-	Sonderzeichen	-	Plus 0
-	a-z	-	0-9
-	A-R	-	Fehlerkennzeichen
-	Minus 0		

- Ein numerischer Vergleich wird durchgeführt, wenn einer oder beide Operanden numerisch sind. Ein numerischer Vergleich wird nur nach Wert durchgeführt, die Länge oder der Typ der Operanden spielt keine Rolle. So werden z.B. eine Variable mit dem Inhalt "00014" und das alphanumerische Literal "14" in einem numerischen Vergleich als gleich erkannt.

Sind beide Operanden Felder, wird ein alphanumerischer Vergleich durchgeführt. Sollen zwei Felder numerisch verglichen werden, muß eins der Felder in einen arithmetischen Ausdruck eingebaut werden.

Beispiel:

```
IF␣(1)␣=␣(2)   ist ein alphanumerischer Vergleich

IF␣(1)␣*␣1␣=␣(2)   ist ein numerischer Vergleich
```

Das Oder-Format

Mehrere Bedingungsabfragen können durch OR miteinander verknüpft werden.

Format:

```
IF␣Vergleich␣OR␣IF␣Vergleich␣[OR␣IF␣Vergleich]...
                Befehl  [;␣Befehl]...
```

Der nächste Befehl wird ausgeführt, wenn mindestens eine der Bedingungen erfüllt ist. Ist keine der Bedingungen erfüllt, werden alle Befehle bis zum nächsten Punkt übergangen.

Beispiel:

```
IF␣(1)␣>␣100␣OR␣IF␣(2)␣>␣900␣PAUSE␣'FEHLER'.
```

Wird ein Operand gegen mehrere andere Operanden verglichen, kann eine abgekürzte Schreibweise gewählt werden.

Beispiel:

```
IF␣(1)␣=␣100␣OR␣=␣900␣PAUSE␣'FEHLER'.
```

Die Vergleichsbedingungen und das Wort OR müssen geschrieben werden.

	Befehlsstruktur	IF

Das Und-Format

Mehrere Bedingungsabfragen können in einem logischen Und-Format verknüpft werden.

Format:

IF␣Vergleich␣IF␣Vergleich␣[IF␣Vergleich]...
Befehl [;␣Befehl]...

Der nächste Befehl wird ausgeführt, wenn alle Bedingungen erfüllt sind. Ist eine der Bedingungen nicht erfüllt, werden alle Befehle bis zum nächsten Punkt übergangen.

Beispiel:

```
IF␣(1)␣=␣100␣IF␣(2)␣=␣900␣PAUSE␣'FEHLER'.
```

Das Wort IF muß immer geschrieben werden.

Das kombinierte Und-/Oder-Format

Mehrere Bedingungsabfragen können zu einem logischen Und-/Oder-Format verknüpft werden. Die Bedingungen werden dabei ohne eine Hierarchie von links nach rechts abgearbeitet.

Beispiel:

IF␣Bedingung 1␣OR␣IF␣Bedingung 2␣IF␣Bedingung 3␣
Befehl [;␣Befehl]...

Bedingung 1	J	J	J	J	N	N	N	N
Bedingung 2	J	J	N	N	J	J	N	N
Bedingung 3	J	N	J	N	J	N	J	N
Kombinierte Bedingung	J	N	J	N	J	N	N	N

J = Bedingung erfüllt, N = Bedingung nicht erfüllt.

	Befehlsstruktur	IF

IF␣Bedingung 1␣IF␣ Bedingung 2␣OR␣IF␣Bedingung 3␣
Befehl [;␣Befehl]...

Bedingung 1	J	J	J	J	N	N	N	N
Bedingung 2	J	J	N	N	J	J	N	N
Bedingung 3	J	N	J	N	J	N	J	N
Kombinierte Bedingung	J	J	J	N	N	N	N	N

J = Bedingung erfüllt, N = Bedingung nicht erfüllt.

5.6.17 INCLUDE

Format:

INCLUDE␣[Dateiname 1]␣IN ␣ Dateiname 2 ␣ USING␣{Literal / Variable / Feld}.

Funktion:

Durch den INCLUDE-Befehl wird ein neuer Schlüssel in den Index eingeordnet und ein entsprechender Verweis zum zugeordneten Datensatz aufgebaut.

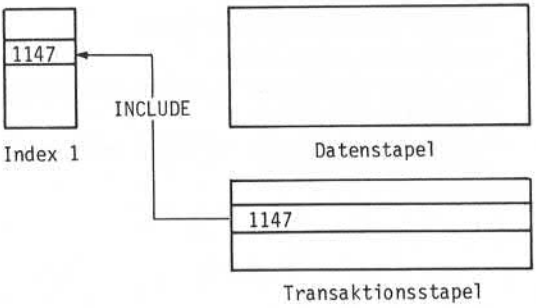
Der INCLUDE-Befehl kann zwei unterschiedliche Operationen ausführen:

- INCLUDE␣IN␣Dateiname␣USING␣{Literal / Variable / Feld}.

  Diese Variante des Befehls verknüpft den eingefügten Schlüssel mit dem gerade aktuellen Satz des Transaktionsstapels. Voraussetzung ist, daß die Indexdatei durch den Befehl OPEN|INC eröffnet wurde.

Beispiel:

```
INCLUDE␣IN␣INDEX1␣USING␣(1).
```



	Befehlsstruktur	INCLUDE

- INCLUDE␣Dateiname 1␣IN␣Dateiname 2␣USING␣{Literal / Variable / Feld}.

  Dieser Befehl verknüpft den eingefügten Schlüssel mit dem Satz des Datenstapels, auf den der Satzzeiger verweist. Dateiname 1 ist der Name der Indexdatei, auf die mit dem letzten GET-Befehl zugegriffen wurde. Dateiname 2 ist der Name des Index, in den der Schlüssel eingefügt werden soll. Dateiname 1 und Dateiname 2 werden in der Regel identisch sein. Soll jedoch in eine Indexdatei mit mehreren Indizes ein Satz eingefügt werden, muß wie im folgenden Beispiel vorgegangen werden:

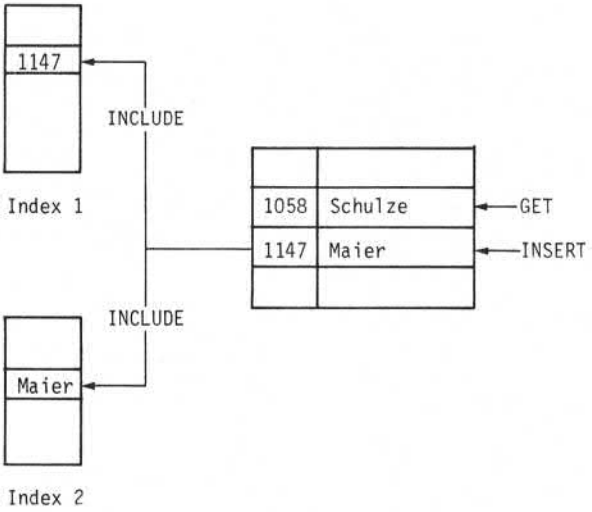
```
GET␣INDEX1␣USING␣SCHL
     ELSE␣GOTO␣!FEHLT.
INSERT␣1␣AFTER␣INDEX1.

eingefügten Satz mit Daten füllen

INCLUDE␣INDEX1␣IN␣INDEX1␣USING␣@1@ .
INCLUDE␣INDEX1␣IN␣INDEX2␣USING␣@2@ .
```

Folgende Operationen werden durch den obigen Programmausschnitt durchgeführt:

- Bestimmten Satz lesen.
- Dahinter einen Leersatz einfügen.
- Leersatz mit Daten füllen.
- Feld 1 des Datensatzes in INDEX1 aufnehmen und mit dem neuen Satz verknüpfen.
- Feld 2 des Datensatzes in INDEX2 aufnehmen und mit dem neuen Satz verknüpfen.

	Befehlsstruktur	INCLUDE

Beispiel



Ist der in den Index eingefügte Schlüssel schon vorhanden, wird er als letzter hinter alle identischen Schlüssel eingefügt.

Achtung! Werden große Mengen von Schlüsseln in einem Index eingefügt, sollten die Schlüssel aufsteigend sortiert sein. Dadurch wird der Vorgang des Einfügens beschleunigt und die Plattenkapazität besser ausgenutzt.

5.6.18 INSERT

Format:

INSERT␣ { numerisches Literal / Variable } ␣ { BEFORE / AFTER } ␣ Dateiname.

Funktion:

Der INSERT-Befehl fügt einen neuen Satz in den Datenstapel einer Indexdatei ein.

Das Format des eingefügten Satzes wird durch das numerische Literal oder die numerische Variable angewählt. Als Literal oder Inhalt der Variablen ist nur eine Ziffer von 0 - 9 zulässig.

Die Formatangabe bezieht sich auf die Eingabeformate, mit denen der Datenstapel angelegt wurde. Besteht eine Indexdatei aus mehreren verbundenen Datenstapeln, gelten die Eingabeformate der einzelnen Datenstapel (Vorsicht!).

Der neue Datensatz kann vor (BEFORE) oder nach (AFTER) dem Datensatz eingefügt werden, auf den der Satzzeiger verweist. Der Programmierer muß dafür sorgen, daß der Satzzeiger auf die richtige Stelle des Datenstapels verweist.

Beachten Sie bitte folgende Punkte:

- Durch den Befehl INSERT wird ein Leersatz gebildet, der anschließend vom Programm mit Daten gefüllt werden muß. Durch den nächsten Dateizugriff oder durch den CLOSE-Befehl (bzw. ohne CLOSE-Befehl bei Beendigung des Programms) wird der neue Satz in den Datenstapel eingefügt.
- Der INSERT-Befehl schafft keine Verbindung zwischen eingefügtem Satz und dem Index der Datei. Diese Verbindung muß - falls erforderlich - mit einem INCLUDE-Befehl hergestellt werden.

	Befehlsstruktur	INSERT

Beispiel:

```
GET␣INDEX1␣USING␣SCHL.
INSERT␣1␣AFTER␣INDEX1.
MOVE␣(1)␣TO␣@1@.
MOVE␣(2)␣TO␣@2@.
MOVE␣(3)␣TO␣@3@.
INCLUDE␣INDEX1␣IN␣INDEX1␣USING␣@1@.
```

5.6.19 LINK

Format:

LINK ␣ { Literal / Feld / Variable } .

Funktion:

Der LINK-Befehl wählt das nächste Eingabeformat an. Die Eingabeformatverkettung des Jobs oder eine Kettfeld-Angabe im Eingabeformat wird übersteuert.

Der LINK-Befehl ist nur im Feldende-Editor zulässig und nur im Eingabe-Modus wirksam.

Die letzte Stelle des Literals oder des Feld- bzw. Variableninhalts ist die Nummer des nächsten Eingabeformats.

Es können mehrere LINK-Befehle nacheinander ausgeführt werden, da der Befehl erst nach dem Verlassen des Satzes wirksam wird. In diesem Fall ist immer der letzte LINK-Befehl gültig.

Beispiel:

Das Programm sieht vor, daß nach Abarbeitung von FMT 4 zu FMT 5 verzweigt wird, wenn Feld 3 = M ist bzw. zu FMT 6, wenn Feld 3 = W ist.

```
WHEN␣NOT␣FMT␣4␣RELEASE.
IF␣(3)␣=␣'M'␣LINK␣5.
IF␣(3)␣=␣'W'␣LINK␣6.
RELEASE.
```

5.6.20 LOAD

Format:

LOAD␣Variable.

Funktion:

Der LOAD-Befehl überträgt den Inhalt eines Stapelsummenakkumulators in eine Variable.

Der Befehl ist nur im Feldende-Editor zulässig.

Die Position der Variablen im DECLARE-Befehl bestimmt die Nummer des Stapelsummenakkumulators.

Beispiel:

```
          DECLARE  A  B  C  D  E.
     Akkumulator:  1  2  3  4  5
```

Typ und Länge der Variablen werden durch den LOAD-Befehl nicht verändert.

5.6.21 MOVE

Format:

MOVE ␣ { Literal / Feld / Variable / Arithmetischer Ausdruck } ␣TO␣ { Feld / Variable } .

Funktion:

Der Inhalt des ersten Operanden wird in den zweiten Operanden übertragen. Dabei wird der Inhalt des zweiten Operanden überschrieben; der erste Operand bleibt unverändert.

Ist der zweite Operand eine Variable, so werden Typ und Länge der Variablen ggfs. durch den MOVE-Befehl verändert (siehe Abschnitt 5.2.4).

Soll eine Variable eine bestimmte Länge erhalten, so kann dies durch den Transport eines alphanumerischen Literals erreicht werden, z.B. MOVE␣'␣␣␣␣␣'␣TO␣A. Eine anschließende Addition von Null auf die Variable wandelt diese in eine numerische Variable der festgelegten Stellenzahl um.

Ist der zweite Operand ein Feld, werden Typ und Länge nicht verändert. Der Feld-Typ wird im MOVE-Befehl grundsätzlich alphanumerisch angenommen.

Die Art der Obertragung kann numerisch oder alphanumerisch sein und ist vom Typ des ersten Operanden abhängig:

- Ist der erste Operand numerisch, erfolgt die Obertragung numerisch. Die Zeichen werden im zweiten Operanden rechtsbündig abgestellt. Nicht belegte Zeichenpositionen werden links mit Nullen aufgefüllt.

	Befehlsstruktur	MOVE

- Ist der erste Operand alphanumerisch, erfolgt die Übertragung alphanumerisch. Die Zeichen werden im zweiten Operanden linksbündig abgestellt. Nicht belegte Zeichenpositionen werden rechts mit Leerzeichen aufgefüllt. Da ein Feld im MOVE-Befehl immer als alphanumerisch gilt, muß, um ein rechtsbündiges Abstellen zu erreichen, das Feld in einen arithmetischen Ausdruck eingebaut werden (z.B. MOVE␣(7)␣*␣1␣TO␣(2)).

Ist der erste Operand länger als der zweite Operand, werden die überschüssigen Zeichen links bzw. rechts abgeschnitten.

Beispiel:

```
MOVE␣(1)␣TO␣TOTAL.
```

Der Inhalt von Feld 1 wird in die Variable TOTAL transportiert. Die Variable hat nach der Transportoperation die gleiche Länge wie Feld 1.

```
MOVE␣(2)␣+␣(3)␣TO␣TOTAL.
```

Die Summe der Felder 2 und 3 wird in die Variable TOTAL transportiert, die Variable ist anschließend 14-stellig.

```
MOVE␣(13:3-5)␣TO␣(7).
```

Die Stellen 3,4 und 5 des Feldes 13 werden in das Feld 7 transportiert und linksbündig abgestellt.

5.6.22 MULTIPLY

Format:

MULTIPLY␣{Literal / Feld / Variable / Arithmetischer Ausdruck}␣TIMES␣Variable.

Funktion:

Der Inhalt des ersten Operanden wird mit dem Inhalt des zweiten Operanden multipliziert. Das Ergebnis wird im zweiten Operanden abgestellt. Der erste Operand wird nicht verändert. Die Länge des zweiten Operanden bleibt erhalten.

Ist das Ergebnis der Multiplikation länger als der zweite Operand, findet ein Oberlauf statt. Der Oberlaufmerker wird gesetzt und kann vom Programm abgefragt werden. Der Oberlauf geht verloren.

Wird ein alphanumerischer Operand verwendet, rechnet das System mit den rechten 4 Bit des EBCDIC-Codes.

Beispiel:

```
Feld 2  = |0|0|0|2|4|
WERT    = |0|0|1|5|9|
MULTIPLY␣(2)␣TIMES␣WERT.
WERT    = |0|3|8|1|6|
```

5.6.23 NOTE

Format:

NOTE␣Kommentar.

Funktion:

Der NOTE-Befehl erlaubt das Einfügen von Kommentaren in das Quellprogramm.

Als Kommentar kann jede beliebige Zeichenfolge mit Ausnahme das Punktes codiert werden. Der Punkt beendet den Kommentar. Der NOTE-Befehl wird nicht compiliert und hat keine Auswirkung auf den Programmablauf.

Beispiel:

```
DECLARE ␣NR.
NOTE␣**␣NR␣=␣KUNDENNUMMER␣** .
```

5.6.24 OPEN

Format:

OPEN␣Dateiname.

OPEN␣Dateiname|UPD.

OPEN␣Dateiname|INC.

Funktion:

Der OPEN-Befehl eröffnet eine Indexdatei zur Verarbeitung durch das Programm. Maximal 4 Indexdateien können gleichzeitig in einem Programm eröffnet werden. Vor der Eröffnung muß die Datei durch einen DEFINE-Befehl definiert worden sein. Eine eröffnete Datei muß erst durch den CLOSE-Befehl geschlossen werden, bevor sie wieder eröffnet werden kann. Es ist daher empfehlenswert, den OPEN-Befehl in die Bedingungsabfrage WHEN␣START einzubauen (z.B. WHEN␣START␣OPEN␣ARTSTAMM).

Der OPEN-Befehl erlaubt 3 verschiedene Arten der Dateieröffnung:

- OPEN Dateiname.

  Diese Eröffnungsart erlaubt nur das Lesen in einer Indexdatei. Das Löschen, Einfügen und Ändern von Datensätzen oder Schlüsseln ist nicht möglich.

- OPEN Dateiname|UPD.

  Erlaubt das Lesen, Löschen, Einfügen und Ändern von Datensätzen und/oder Schlüsseln.

- OPEN Dateiname|INC.

  Erlaubt die gleichen Funktionen wie OPEN|UPD. Zusätzlich können Sätze des Transaktionsstapels mit dem Index verknüpft werden. Durch diese Eröffnungsart wird der Transaktionsstapel an die Indexdatei gebunden und kann erst gelöscht werden, nachdem die Indexdatei gelöscht wurde (Vorsicht!).

	Befehlsstruktur	OUTPUT

5.6.25 OUTPUT

Format:

Funktion:

Der OUTPUT-Befehl führt folgende Operationen aus:

- Ausgabe von Feldern, Literalen, Variablen und Befehlsergänzungen auf Band, Drucker oder Datenfernübertragung.
- Reformatierung von Daten durch Masken und Befehlsergänzungen.
- Ausgabe einer Fehlerliste auf Platte im Stapelende-Editor.

Durch jeden OUTPUT-Befehl wird ein Datensatz ausgegeben (Ausnahme <DEFER>). Auf dem Drucker wird durch den OUTPUT-Befehl mindestens 1 Zeilenschaltung ausgeführt.

Feld- und Variableninhalte können durch eine Maskenangabe für die Ausgabe aufbereitet werden. Die Maskenangabe erfolgt direkt hinter der Feldnummer bzw. dem Variablennamen. Der Maskenangabe geht ein senkrechter Strich (|) voraus.

z.B.: OUTPUT␣(2)|PK.
OUTPUT␣TOTAL|LZ.

	Befehlsstruktur	OUTPUT

Folgende Maskenangaben sind möglich:

- |PK
- |LS
- |LZ
- |TS
- |TZ
- |SG
- |'Maske'

5.6.25.1 |PK (packed decimal format)

Die Zeichen werden dezimal gepackt. Pro Byte werden zwei numerische Zeichen verschlüsselt, das Vorzeichen wird im letzten Halbbyte abgestellt.

z.B.: |F,1| |F,2| |F,3| |F,4| ungepackt

|0,1| |2,3| |4,F| gepackt

5.6.25.2 |LS (Truncate all leading spaces)

Führende Leerzeichen werden unterdrückt; das Feld wird in der Länge auf die verbleibenden Zeichen reduziert.

5.6.25.3 |LZ (Truncate all leading zeroes)

Führende Nullen werden unterdrückt (Längenreduzierung wie oben).

5.6.25.4 |TS (Truncate all trailing spaces)

Nachfolgende Leerzeichen werden unterdrückt (Längenreduzierung wie oben).

5.6.25.5 |TZ (Truncate all trailing zeroes)

Nachfolgende Nullen werden unterdrückt (Längenreduzierung wie oben).

Bei den Angaben |LS, |LZ, |TS und |TZ wird die echte Anzahl Zeichen des entsprechenden Feldes bzw. der entsprechenden Variablen ausgegeben, d.h. abhängig von der Spezifikation entfallen Nullen oder Blanks. Somit ergibt sich bei der Ausgabe eine variable Feld- bzw. Variablenlänge.

5.6.25.6 |SG

Durch die Maskenangabe |SG wird die letzte Stelle eines Feldes mit einem positiven OVERSIGN versehen.

5.6.25.7 |'Maske'

Durch die OUTPUT-Befehlsergänzung |'Maske', die ohne Berücksichtigung der beiden Hochkommas eine maximale Länge von 20 Zeichen haben darf, sind bei der Ausgabe numerischer Werte die folgenden Darstellungsvarianten möglich:

|'----' Die Anzahl der Unterstreichungsstriche legt die Anzahl der Zeichen bei der Ausgabe fest. Beinhaltet das auszugebende Feld mehr Zeichen als durch die Maske festgelegt wurde, stellt das System nur die rechtsbündig stehenden Zeichen ab.

	Befehlsstruktur	OUTPUT

|'---Ø---' Wird innerhalb der Maske eine Null (Ø) angegeben, so wird mit Vornullenunterdrückung gearbeitet. Die Stelle, auf der die Ø steht, wird noch mit in die Vornullenunterdrückung einbezogen.

|'---*---' Wird innerhalb der Maske ein Stern (*) angegeben, so werden statt führender Nullen Sterne ausgegeben. Dies gilt bis einschließlich der Stelle, auf der der Stern steht.

|'---,---' Wird innerhalb der Maske ein Komma (,) angegeben, so wird dieses Komma bei der Ausgabe eingeblendet.

|'---.---' Wird innerhalb der Maske ein Punkt (.) angegeben, kann folgendes damit erzielt werden:

- Bei Vornullenunterdrückung wird die Punkt-Angabe, sofern noch keine Ziffer ≠ 0 aufgetreten ist, ignoriert. Stattdessen wird ein Blank ausgegeben.
  Wurden Ziffern ≠ 0 gefunden, wird der Punkt ausgegeben.

- Wird ohne Vornullenunterdrückung gearbeitet und steht z.B. links des "Tausender"-Punktes noch eine Ziffer ≠ 0 (Beispiel: 0003.415.678), wird der Punkt eingeblendet.

- Wird ohne Vornullenunterdrückung gearbeitet und steht z.B. links des "Tausender"-Punktes keine Ziffer ≠ 0 (Beispiel: 0000415.678), wird statt des Punktes eine Null ausgegeben.

|'----CR'
oder
|'-----' Das Symbol "CR" (credit" oder "-" (Minus) wird bei negativem Feldinhalt ausgegeben. Bei positivem Feldinhalt werden diese Symbole durch Leerzeichen ersetzt.

Beispiele:

		Ausgabe	
Maske	± Feldinhalt	+Daten	-Daten
|'---0,-- '	000005	,05	,05
|'--0-,-- '	000005	0,05	0,05
|'---*,-- '	000005	****,05	****,05
|'---,--- '	13560	135,60Ø	135,60-
|'---,--CR|	13560	135,60ØØ	135,60CR
|'-.---,--'	176342	1.763,42	1.763,42
|'-.--*,--'	000005	*****,05	*****, 05
|LZ	0000175	175	17N

5.6.26 OUTPUT-Befehlsergänzungen

Die folgende Befehlsergänzungen werden in die Symbole "kleiner als" (<) und "größer als" (>) eingeschlossen (z.B. OUTPUT <TOP>, OUTPUT␣<RWND>) und können in beliebiger Anzahl nach dem OUTPUT-, AUDIT- PAUSE- SHOW- SORT- und TYPE-Befehl codiert werden.

5.6.26.1 <ALL>

Durch die ALL-Anweisung können mehrere Felder gleichzeitig ausgegeben werden.

- <ALL>

  gibt alle Felder des aktuellen Satzes im Stapel aus.

- <ALL␣n>

  gibt alle Felder ab dem durch n angegebenen Feld bis zum Ende des aktuellen Satzes im Stapel aus.

- <ALL␣n-m>

  gibt ab dem durch n angegebenen Feld bis zu dem durch m angegebenen Feld alle Felder des aktuellen Satzes im Stapel aus.

- <@ALL>

  gibt alle Felder der aktuellen Satzes der Indexdatei aus.

- <@ALL␣n>

  gibt alle Felder ab dem durch n angegebenen Feld bis zum Ende des aktuellen Satzes der Indexdatei aus.

- <@ALL␣n-m>

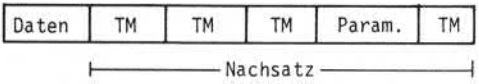
  gibt ab dem durch n angegebenen Feld bis zu dem durch m angegebenen Feld alle Felder des aktuellen Satzes der Indexdatei aus.

5.6.26.2 <APPEND>

Durch die OUTPUT-Ergänzung APPEND wird die Möglichkeit einer Bandfortschreibung gegeben. Das System überschreibt bei fester Blocklänge mögliche Füllzeichen und stellt an diese Stelle neue Daten; bei variabler Blocklänge wird der Block bis zur maximalen Blocklänge mit Sätzen aufgefüllt.

Sowohl bei fester als auch bei variabler Blocklänge werden in jedem Fall nur ganze Sätze berücksichtigt, d.h. der Block wird nur mit ganzen Sätzen aufgefüllt.

Die Bandfortschreibung durch den APPEND-Befehl wird vom System automatisch durchgeführt. Ermöglicht wird dies durch das Einlesen eines Nachsatzes, der aus 5 Blöcken besteht und generell nach jeder Bandausgabe hinter den Daten abgestellt wird. Diese Blöcke bestehen aus 3 Bandmarken, 1 Parameterblock und wiederum einer Bandmarke. Die Parameter enthalten u.a. Satzzähler, Blockzähler, Zeichenzähler. Das automatische Schreiben dieser 5 Blöcke nach dem letzten Datenblock wird in der weiteren Beschreibung als Nachsatz bezeichnet.



Nach der Ausgabe einer Banddatei sowie des internen Nachsatzes setzt das System das Magnetband automatisch um den Nachsatz, also um 5 Blöcke zurück, d.h. hinter den letzten ausgegebenen Datenblock. An dieser Stelle kann nun der Benutzer normal weiterarbeiten, also eine neue Datei schreiben oder das Magnetband zurückspulen.

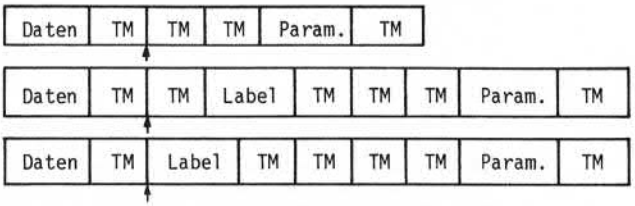
Wird eine Bandfortschreibung eingeleitet, so muß

1. das Magnetband auf die erste Bandmarke vorgesetzt

   und

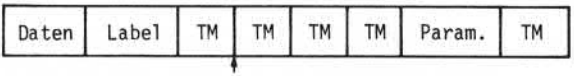
2. das Band entweder über die Supervisorfunktion "Block zurücksetzen" oder den Befehl <BSP n> unmittelbar hinter den letzten Datenblock zurückpositioniert werden. Die Abkürzung TM der Abbildungen bedeutet Tapemark = Bandmarke.

(Das Band befindet sich mit der durch einen Pfeil bezeichneten Stelle am Schreib-/Lesekopf.)



Programmierung für die obigen Beispiele:

WHEN␣START␣OUTPUT␣<BSP␣1>␣<APPEND>.



Programmierung für das letzte Beispiel:

WHEN␣START␣OUTPUT␣<BSP␣2>␣<APPEND>.

Wichtig ist, daß einem Label-Block nicht mehr als 2 Bandmarken vorausgehen dürfen.

Da nach jeder Bandausgabe automatisch ein Nachsatz geschrieben wird, darf speziell bei Bandfortschreibung keine Bandmarke über die Supervisor-Funktion gesetzt werden.

Bei falscher Anwendung der Befehlsergänzung APPEND erfolgt eine Fehlermeldung (siehe Systemfehlermeldungen).

5.6.26.3 < BATCH >

Durch diese Anweisung wird der Stapelname in den Ausgabesatz eingefügt. Es werden dabei 10 Stellen belegt, auch wenn der Name aus weniger als 10 Zeichen besteht.

5.6.26.4 < BLK␣n >

Durch die BLK-Anweisung wird der Inhalt des physikalischen Blockzählers des Systems auf Band ausgegeben. "n" gibt die Anzahl der Stellen an, in denen der Blockzähler abgestellt wird. "n" kann max. mit 5 angegeben werden. Ist "n" = 0, wird die Anweisung ignoriert.

Ist der Blockzähler stellenmäßig größer als die in der BLK-Anweisung angegebene Stellenanzahl, kann er nicht in der vollen Länge dargestellt werden.

5.6.26.5 < BSP␣nnnn >

Durch diese Anweisung wird das Magnetband um die unter "nnnn" angegebene Anzahl von Blöcken zurückgesetzt. "nnnn" kann max. 2.047 sein.

5.6.26.6 < COUNT␣xxxx >

Mit Hilfe der COUNT-Anweisung wird die Anzahl Zeichen in einem Ausgabesatz gezählt und ausgegeben. Der COUNT kann dezimal (D) oder binär (B) durchgeführt werden.

Die Darstellung erfolgt auf max. 4 Stellen mit führenden oder nachfolgenden Füllzeichen (S).

Beispiele:

DDSS Die Anzahl der Zeichen im Satz wird mit 2 Dezimalstellen und 2 nachfolgenden Füllzeichen ausgegeben.

SSBB Die Anzahl der Zeichen im Satz wird mit 2 führenden Füllzeichen und 2 Binärstellen ausgegeben.

SDDD Die Anzahl der Zeichen im Satz wird mit einer führenden Null und 3 Dezimalstellen ausgegeben.

Die Füllzeichen (S) zwischen den Zeichen (D oder B), ebenso wie die Kombination von D und B sind nicht erlaubt, z.B. DSDS, BDSS, BSBB.

Die verwendeten Füllzeichen werden bei der Ausgabe des Stapel definiert (siehe Ausgabeparameter "Füllzeichen" im Standard-Job).

5.6.26.7 <DATE␣x>

Durch die DATE-Anweisung wird das vom Benutzer über die Supervisorfunktion sechsstellig eingegebene Datum ausgegeben. "x" legt fest, welches Zeichen zwischen Tag (TT), Monat (MM) und Jahr (JJ) erscheinen soll.

"x" kann jedes beliebige Tastaturzeichen sein. Eine Ausnahme bildet der Unterstreichungsstrich. Wird er in der DATE-Anweisung angegeben, wird er als Blank dargestellt.

Beispiel:

<DATE␣.> ergibt TT.MM.JJ
<DATE␣/> ergibt TT/MM/JJ

Fehlt die Angabe "x", wird das Datum sechsstellig ausgegeben (TTMMJJ).

5.6.26.8 <DEFER>

Die DEFER-Anweisung verhindert die Ausgabe des Datensatzes. Die Ausgabe der Daten wird solange verzögert, bis ein weiterer OUTPUT-Befehl gegeben wird oder der Ausgabepuffer gefüllt ist (vorausgesetzt, dieser OUTPUT-Befehl enthält nicht eine nochmalige DEFER-Anweisung).

Der DEFER-Befehl ist sinnvoll, wenn z.B. aus mehreren Eingabesätzen ein Ausgabesatz zusammengestellt werden soll.

Achtung! Nach <DEFER> beginnt die SKIP-Anweisung ab der letzten Position zu zählen.
(gilt nur für den TYPE-Befehl).

5.6.26.9 <EOF>

Die EOF (end of file)-Anweisung bewirkt, daß eine auf Magnetband ausgegebene Datei durch Schreiben einer Bandmarke abgeschlossen wird.

Wurde für die Ausgabe feste Blocklänge spezifiziert und der letzte Ausgabeblock nicht bis zur vollen Länge mit Sätzen beschrieben, füllt das System bei der EOF-Anweisung zunächst den Block mit den im Standard-Job angegebenen Füllzeichen auf und schreibt anschließend die Bandmarke.

Wurde für die Ausgabe variable Blocklänge spezifiziert, schließt das System bei der EOF-Anweisung die Bandmarke unmittelbar an die letzte Datenausgabe an.

Bei der Ausgabe auf Drucker wird eine EOF-Anweisung ignoriert.

Bei der Ausgabe auf Datenfernübertragungsleitung bewirkt dies den Abbruch der Datenübertragung.

Achtung! Bei Beendigung einer Bandausgabe durch das Programm werden vom System automatisch 3 Bandmarken geschrieben. Die EOF-Anweisung muß daher nur codiert werden, wenn mehrere Dateien auf ein Band geschrieben und durch Bandmarken getrennt werden sollen.

5.6.26.10 <FMT>

Die FMT-Anweisung fügt die Nummer des Eingabeformats, mit dem der Satz erfaßt wurde, in die Ausgabe ein.

5.6.26.11 <HEX␣xx>

Das mit "xx" angegebene Zeichen wird in einem Byte ausgegeben. Eine Ausgabe-Codewandlung findet nicht statt.
Bei DFO ist die Ausgabe von HEX-Zeichen nicht zulässig!

5.6.26.12 <JOB>

Der Name des Standard-Jobs wird mit 8 Zeichen in den Ausgabesatz eingefügt. Ist der Name kürzer als 8 Stellen, wird mit Leerzeichen aufgefüllt.

5.6.26.13 <KEY>

Diese Befehlsergänzung erlaubt den Zugriff auf den Indexschlüssel des zuletzt gelesenen Dateisatzes. Voraussetzung ist, daß der Schlüssel im EBCDIC-Code aufgebaut wurde.
Die Verwendung dieser Befehlsergänzung ist z.B. sinnvoll beim sequentiellen Leser einer Indexdatei (GET NEXT, GET PRIOR), wenn der Schlüssel nicht im Datensatz enthalten ist.

5.6.26.14 <LABEL>

Die LABEL-Anweisung kennzeichnet einen Satz als Kennsatz, d.h. bei dem mit LABEL versehenen Satz handelt es sich nicht um einen Datensatz. Die LABEL-Anweisung muß die erste Angabe nach dem OUTPUT-Befehl sein. Der Kennsatz wird unabhängig vom Blockungsfaktor geschrieben und nicht im Satz- und Blockzähler berücksichtigt.

5.6.26.15 <LF>

Die <LF> -Anweisung bewirkt einen Zeilenvorschub auf dem Drucker. Sollen mehrere Zeilenvorschübe durchgeführt werden, muß die Angabe <LF> entsprechend oft gemacht werden.

Bei einer Ausgabe auf Band wird die Anweisung ignoriert.

Bemerkung:

Der OUTPUT-Befehl selbst bewirkt bereits einen Zeilenvorschub. Bei der Angabe "OUTPUT␣ <LF> ." werden folglich 2 Zeilenvorschübe durchgeführt.

5.6.26.16 <PAGE␣nn>

Die PAGE-Anweisung legt die Blatthöhe des Arbeitsplatzdruckers fest. Der Zeilenzähler des Druckers wird durch die PAGE-Anweisung auf Null gesetzt. Die Blatthöhe bleibt gesetzt, bis eine neue PAGE-Anweisung gegeben oder der Bildschirm abgeschaltet wird. Ohne PAGE-Anweisung ist die Blatthöhe auf 66 Zeilen eingestellt.

5.6.26.17 <REC␣n>

Durch die REC-Anweisung wird der Inhalt des physikalischen Satzzählers geschrieben. "n" gibt dabei die Anzahl der Stellen an. Der Satzzähler kann auf max. 5 Stellen abgestellt werden; ist "n" = 0, wird die Anweisung ignoriert.

5.6.26.18 <RWND>

Durch die RWND-Anweisung wird das Magnetband zurückgespult. Bei der Ausgabe auf Drucker ignoriert das System die Anweisung.

5.6.26.19 <SKIP␣nnnn>

Die SKIP-Anweisung fügt Blanks in den Ausgabesatz ein, beginnend von der momentanen Zeichenposition bis zu der Zeichenposition, die durch "nnnn" spezifiziert wurde.

"nnnn" darf max. 2.047 betragen.

Achtung! Nach der Anweisung DEFER beginnt SKIP ab der letzten Position zu zählen.
(gilt nur für den TYPE-Befehl).

5.6.26.20 <TIME␣x>

Die TIME-Anweisung erlaubt den Zugriff auf die Systemzeit. Die Systemzeit wird in der Supervisorebene gesetzt und automatisch aktualisiert.
Die Uhrzeit wird in folgendem Format dargestellt: Stunden, Stunden/Minuten, Minuten/Sekunden, Sekunden.
Das mit "x" angegebene Zeichen kann wie folgt definiert werden:

- Leerzeichen: Die Zeit wird im 6-Zeichen-Format (hhmmss) ausgegeben.
- "␣" : Die Zeit wird im 8-Zeichen-Format (hh mm ss) ausgegeben.
- Jedes beliebige Tastaturzeichen (Ausnahme "␣"): Das angegebene Zeichen wird zwischen Stunden/Minuten und Minuten/Sekunden eingefügt (hhxmmxss).

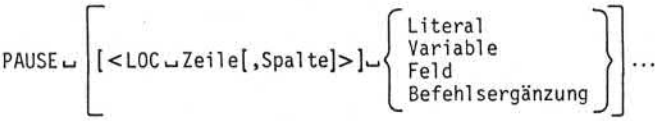
5.6.26.21 <TOP>

Die TOP-Anweisung bewirkt einen Vorschub am Drucker auf TOP-OF-FORM (Blattanfang).

Achtung! Der Blattanfang wird beim Systemdrucker durch den Vorschubstreifen festgelegt. Beim Arbeitsplatz-Drucker wird der Blattanfang durch die PAGE-Anweisung bestimmt.

5.6.27 PAUSE

Format:



Funktion:

Der PAUSE-Befehl stoppt den Programmablauf, zeigt eine Nachricht auf dem Bildschirm an und schaltet den Fehlerton ein. Nach Betätigung der RESET-Taste wird der Fehlerton ausgeschaltet und das Programm fortgesetzt.

Die Angabe von Zeile und Spalte legt die Position der Nachricht auf dem Bildschirm fest. Zeile und Spalte können als numerische Literale oder Variable codiert werden. Die Angabe der Variablen erlaubt das Anzeigen von Nachrichten auf verschiedenen Bildschirmpositionen durch nur einen Befehl (z.B. in einem Unterprogramm zur Fehleranzeige). Die Angabe [<LOC␣Zeile [,␣Spalte]>] kann mehrfach in einem Befehl auftreten. Dabei braucht keine bestimmte Reihenfolge der Bildschirmpositionen eingehalten werden.

Bei Festlegung der Bildschirmposition ist zu beachten, daß Zeile 1 die Statuszeile und Zeile 2 die Fehlerzeile ist. Beide Zeilen werden ggfs. überschrieben (Vorsicht!). Wird keine Positionsangabe codiert, wird die Nachricht auf dem 480-Zeichen Bildschirm in der Fehlerzeile (Zeile 2) und auf dem 1920-Zeichen Bildschirm in Zeile 1 ab Position 41 angezeigt.
Die Nachricht kann aus einer beliebigen Kombination von Literalen, Variablen und Feldern bestehen. Ein Teil der im Abschnitt 5.6.26 beschreibenen Befehlsergänzungen können ebenfalls verwendet werden.

Als Befehlsergänzungen sind <CRT> , <BATCH> , <JOB> , <DATE␣x> , <ALL> und @ALL@ zulässig. Die Anweisungen <TOP> und <LF> haben Sonderfunktionen. PAUSE <TOP> löscht den Bildschirm ab Zeile 3; PAUSE <LF> bewirkt einen Roll-up um eine Zeile.

Die Befehlsergänzung < CRT␣480 > setzt die Bildschirmgröße auf 480 Zeichen, < CRT␣1920 > setzt die Bildschirmgröße auf 1920 Zeichen (nur möglich, wenn ein Bildschirmplatz Modell 3 mit 1920 Zeichen Kapazität benutzt wird.
Die Befehlsergänzung < CRT > ist in allen Editor-Programmen zulässig.

Felder und Variable können mit Maskenangaben (siehe Punkt 5.6.25.1 - 5.6.25.7) versehen werden. Ein Literal innerhalb der PAUSE-Befehls darf max. 40 Stellen lang sein. Soll ein längeres Literal angezeigt werden, muß es in mehrere Literale unterteilt werden.

	Befehlsstruktur	PAUSE

Eine Nachricht, die aus mehr als 200 Zeichen besteht, erzeugt einen Fehler zur Laufzeit des Programms (Vorsicht!).

Ein PAUSE-Befehl ohne Nachricht löscht die Fehlerzeile des Bildschirms.

Folgt eine PAUSE-Anzeige einer anderen, werden nur soviele Positionen der alten Anzeige überschrieben, wie die neue Anzeige stellenmäßig benötigt. Das bedeutet: Folgt eine kurze Anzeige einer langen Anzeige, wird die kurze Anzeige zusammen mit dem Ende der langen Anzeige eingeblendet. Daher ist es empfehlenswert, im PAUSE-Befehl die für die Anzeige nicht benötigten Positionen mit Blanks aufzufüllen.

Beispiel:

```
IF␣ DIFF␣ ≠␣ 0
    PAUSE ␣'DIFFERENZ␣='␣ DIFF|'-.-0-,--¯'␣'DM'␣18'␣'.
```

Im Überprüf-Modus und in der ABL-Funktion wird der PAUSE-Befehl ignoriert.

5.6.28 PERFORM

Format:

PERFORM Label.

Funktion:

Durch den PERFORM-Befehl verzweigt das Hauptprogramm in ein Unterprogramm, das mit dem im PERFORM-Befehl angegebenen Label beginnt.

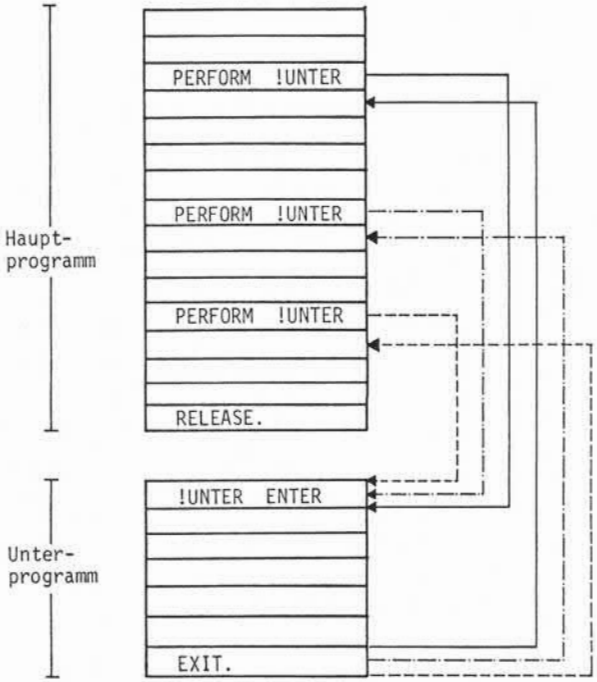
Der Label kann aus 2 bis 9 Zeichen bestehen. Das erste Zeichen muß ein Ausrufungszeichen sein, danach folgt zwingend ein Buchstabe. Der Rest des Labels kann aus Buchstaben oder Ziffern bestehen. Die Anzahl der Label in einem Programm ist nicht begrenzt. Nach Abarbeitung des Unterprogramms wird das Hauptprogramm mit dem Befehl fortgesetzt, der dem PERFORM-Befehl folgt.

Jedes Unterprogramm muß unmittelbar nach dem Label mit dem Wort ENTER beginnen. Der erste Befehl des Unterprogramms schließt sich ohne Punkt an. Am Ende des Unterprogrammes wird nach dem Punkt des letzten Befehls das Wort EXIT. angegeben. Dadurch wird das Unterprogramm verlassen und im Hauptprogramm weitergearbeitet.

Neben dem EXIT-Befehl kann ein Unterprogramm auch durch den POSITION-Befehl verlassen werden.

In der Regel wird eine Unterroutine verwendet, um mehrfach benötigte Programmfunktionen nur einmal codieren zu müssen.

	Befehlsstruktur	PERFORM



Achtung! In einem linearen Programmablauf ist es unzulässig, ein Unterprogramm ohne den PERFORM-Befehl anzusprechen. Dementsprechend muß vor jedem Unterprogramm ein RELEASE, GOTO, BYPASS, POSITION, EXIT oder STOP codiert werden.

Ein RELEASE␣ELSE- oder BYPASS␣ELSE-Befehl wird vom Programm vor einem Unterprogramm nicht akzeptiert. Vor dem Unterprogramm muß ein unbedingter Befehl stehen.

Es ist nicht zulässig, den PERFORM-Befehl innerhalb eines Unterprogramms zu geben. Das System erkennt in diesem Fall einen Fehler. Es kann nicht in Unterprogramme eines anderen Programms verzweigt werden.

	Befehlsstruktur	PERFORM

Beispiel:

```
PERFORM ␣!MULTI.
```

```
RELEASE.
!MULTI␣ ENTER
MOVE␣ MENGE␣*␣PREIS␣*␣RABATT␣ TO␣ WERT.
DIVIDE␣ 100␣ INTO␣ WERT.
OUTPUT␣<SKIP␣ 82>␣WERT|'--.-0-,--'.
EXIT.
```

5.6.29 POSITION

Format:

POSITION␣[FELD] .

Funktion:

Der POSITION-Befehl ist ausschließlich im Feldende-Editor wirksam. Er ermöglicht eine vom Programm gesteuerte Cursorpositionierung im Eingabesatz, die sonst nur manuell über die Tastatur vorgenommen werden kann.

- Wird im POSITION-Befehl keine weitere Angabe gemacht, positioniert der Cursor im Eingabe-Modus auf die 1.Stelle des 1.Feldes des gerade in Bearbeitung befindlichen Satzes, im Prüf- und Update-Modus auf das erste spezifizierte Prüf- bzw. Update-Feld des aktuellen Satzes.

- Wird im POSITION-Befehl eine Feldangabe (Feld-Nr.) gemacht, positioniert der Cursor auf die 1.Stelle des gewünschten Feldes.

- Wird ein POSITION-Befehl gegeben, nachdem durch einen BACK-/FORWARD-Befehl zurück- bzw. vorpositioniert wurde, verändert sich der Bildschirminhalt; es wird der Satz angezeigt, der durch den BACK-/FORWARD-Befehl erreicht wurde.

Der POSITION-Befehl wirkt zusätzlich wie ein RELAESE-Befehl, d.h. er beendet den Editorlauf.

Wird im Eingabe-Modus auf ein bereits erfaßtes Feld zurückpositioniert, müssen alle nachfolgenden Felder erneut eingegeben werden. Eine nochmalige Eingabe der bereits erfaßten Daten kann allerdings durch Betätigen der LOC RET (Location Return)-Taste umgangen werden.

Wird im Eingabe-Modus auf ein noch nicht erfaßtes Feld positioniert, werden alle übersprungenen Felder mit Füllzeichen (Parameter 14 im Eingabeformat) aufgefüllt und auf dem Bildschirm angezeigt.

Im Prüf- und Update-Modus müssen die Prüf- und Update-Felder erneut eingegeben werden, sofern für diese Felder im Eingabeformat eine entsprechende Spezifikation (wie z.B. Prüfen durch Neueingabe oder Abhängigkeitsprüfung) gemacht wurde.

	Befehlsstruktur	POSITION

Ein wesentlicher Vorteil des POSITION-Befehls liegt darin, daß der Cursor z.B. unmittelbar nach dem ersten vom Editor-Programm erkannten Fehler in das fehlerhafte Feld positioniert. Die Bedienungskraft kann ohne Suchen und Zurückpositionieren korrigieren.

Bemerkungen:

- Der POSITION-Befehl ist für alle Modi - mit Ausnahme des Oberprüfmodus anwendbar.
- Bei der Durchführung der ABL-Funktion wird der POSITION-Befehl ignoriert.
- Die Indizierung der Feldangabe ist möglich.
- Im Eingabe-Modus kann mit dem POSITION-Befehl über Felder, in die noch keine Eingaben erfolgt sind, hinwegpositioniert werden.
- Die Angabe eines Teilfeldes ist möglich, aber nicht sinnvoll, da nur auf die erste Stelle des Feldes positioniert wird.

Position im Oberprüf-Modus:

Der Oberprüf-Modus überprüft die Daten gemäß dem Eingabeformat. Der Durchlauf des Feld-/Satzende-Editors ist damit gewährleistet. Hat der Feld-/Satzende-Editor einen POSITION-Befehl, so wird dieser nur als RELEASE-Befehl behandelt, denn ein Positionieren in ein bereits behandeltes Feld ist nicht möglich. Dies könnte also ein Nichtbeendendes Editors zur Folge haben.

Beispiel:

Im folgenden Programmierbeispiel soll die Bedienungskraft Feld 3 (Nettopreis) und Feld 4 (Bruttopreis) neu eintasten, wenn der Inhalt des Feldes 4 kleiner ist als der Inhalt des Feldes 3:

```
IF␣(4)␣>␣(3)␣OR␣=␣(3)␣GOTO␣!ENDSU.

SHOW␣'NETTOPREIS␣ODER␣BRUTTOPREIS␣FALSCH'.

POSITION␣(3).

!ENDSU␣ADD␣(4)␣TO␣ENDSUM.
```

Nach dem POSITION-Befehl wird der unter der Adresse !ENDSU angegebene Additionsbefehl nicht ausgeführt, da der POSITION-Befehl wie ein RELEASE-Befehl wirkt.

5.6.30 PROTECT

Format:

PROTECT␣{Feld / Teilfeld}␣[ELSE␣Befehl[;␣Befehl]...].

Funktion:

Der PROTECT-Befehl dient speziell zum Sperren von Datensätzen für andere Bildschirmarbeitsplätze.

Der PROTECT-Befehl führt - abhängig von der ersten Stelle des Feldes oder Teilfeldes - zwei Operationen aus:

- Ist die 1.Stelle kein Fehlerkennzeichen, so wird dieses gesetzt.
- Ist die 1.Stelle ein Fehlerkennzeichen, so wird der ELSE-Zweig des Befehls durchlaufen. Ist kein ELSE-Zweig codiert, wird das Programm solange unterbrochen, bis das Fehlerkennzeichen nicht mehr gesetzt ist.

Beispiel:

```
GET␣XDAT␣USING␣SCHL
     ELSE␣RELEASE.
!A100
PROTECT␣@12@
     ELSE␣SHOW␣'SATZ␣IST␣GESPERRT';
     GOTO␣!A100.
        MOVE␣(2)␣TO␣@2@.
        MOVE␣(3)␣TO␣@3@.
CLEAR␣@12@.
```

	Befehlsstruktur	PROTECT

Wird ein Dateisatz von mehreren Bildschirmarbeitsplätzen gleichzeitig gelesen und der erste Bildschirmarbeitsplatz erreicht den Befehl PROTECT, so wird dieser Satz für alle anderen Bildschirmarbeitsplätze gesperrt, bis er durch den Befehl CLEAR freigegeben wird.

5.6.31 RELEASE

Format:

RELEASE␣[ELSE␣Befehl[;␣Befehl]...].

Funktion:

Der RELEASE-Befehl beendet den Ablauf eines Programms. Folgende Funktionen werden durch RELEASE ausgeführt:

- Der Befehl beendet den Programmzyklus und verzweigt auf den ersten Befehl des Programms.
- Im Feldende-Editor kehrt das System in den Eingabe-Modus zurück und die Bedienungskraft kann das nächste Feld erfassen bzw. im Fehlerfall Korrekturen vornehmen.
- Im Stapelende-Editor, Ausgabe und Sortierprogramm wird der nächste Satz des Stapels verarbeitet. Wurde der letzte Satz des Stapels verarbeitet, durchläuft das System den ELSE-Zweig des Befehls. Ist der ELSE-Zweig nicht codiert, wird das Programm beendet. Im Feldende-Editor ist die Codierung des ELSE-Zweiges nicht sinnvoll.

Werden mehrere Stapel in einem Programm verarbeitet (Multi-Batch) wird der ELSE-Zweig nach dem letzten Satz des letzten Stapels durchlaufen.

Wird nach der Verarbeitung des letzten Satzes noch ein OUTPUT-Befehl gegeben, so erhält der letzte Stapel den Status "ausgegeben", unabhängig davon, ob Daten ausgegeben wurden oder nicht.

Beispiel:

```
DECLARE␣SUMME.
ADD␣(2)␣TO␣SUMME.
RELEASE
    ELSE␣SHOW␣SUMME|LZ;
    STOP.
```

5.6.32 REMOVE

Format:

```
REMOVE ␣Dateiname ␣ [ USING ␣ { Variable } ] ␣
                             { Feld     }
                             { Literal  }

        ELSE ␣ Befehl [;␣Befehl]...
```

Funktion:

Der REMOVE-Befehl entfernt einen Schlüssel aus dem Index der angegebenen Datei. Abhängig von der Angabe USING werden unterschiedliche Operationen ausgeführt:

- REMOVE ␣ Dateiname ␣ USING ␣ { Variable } ...
                               { Feld   }
                               { Literal }

  Diese Variante des Befehls entfernt den ersten Schlüssel, der gleich dem nach USING abgegebenen Suchargument ist. Wird kein gleicher Schlüssel gefunden, durchläuft das System den ELSE-Zweig des Befehls.

  Indexzeiger und Satzzeiger sind nach Ausführung des Befehls REMOVE Dateiname USING ... zerstört.

- REMOVE ␣ Dateiname ...

  Der Schlüssel, auf den der Indexzeiger verweist, wird aus dem Index entfernt. Das bedeutet, daß vor Ausführung des REMOVE-Befehls ein gültiger Satz mit dem GET-Befehl gelesen werden muß. Indexzeiger und Satzzeiger werden nicht verändert.

Wird ein Satz aus dem Datenbereich gelöscht, entfernt das Betriebssystem automatisch alle vorhandenen Verkettungen mit Indizes. Der REMOVE-Befehl muß also nur verwendet werden, wenn der Schlüssel eines ungelöschten Datensatzes aus dem Index entfernt werden soll.

	Befehlsstruktur	REMOVE

Beispiel:

```
GET␣XDAT␣USING␣SCHL
    ELSE␣RELEASE.

REMOVE␣XDAT
    ELSE␣STOP.
```

Nach Ausführung des Programms ist nur der Schlüssel gelöscht. Der Datensatz bleibt bestehen.

Wird ein Schlüssel durch den Befehl REMOVE Dateiname USING ... gelöscht, muß unbedingt beachtet werden, daß anschließend auf den zugehörigen Satz des Datenstapels nicht zugegriffen werden kann, weil der Satzzeiger zerstört ist.

Beispiele für falschen Programmaufbau:

```
REMOVE␣XDAT␣USING␣SCHL
    ELSE␣RELEASE.
DELETE␣XDAT.
```

oder:

```
REMOVE␣XDAT␣USING␣SCHL
    ELSE␣RELEASE.
ACCEPT␣SCHL.
INCLUDE␣XDAT␣IN␣XDAT␣USING␣SCHL.
```

5.6.33 SET

Format:

SET␣ Variable.

Funktion:

Der Befehl SET überträgt den Inhalt der Variablen in einen Stapelsummenakkumulator.

Der Befehl ist nur im Feldende-Editor zulässig. Die Position der Variablen im DECLARE-Befehl bestimmt die Nummer des Stapelsummenakkumulators.

Beispiel:

```
            DECLARE   A  B  C  D  E.
        Akkumulator:  1  2  3  4  5
```

5.6.34 SHOW

Format:

SHOW␣[<LOC␣Zeile[,Spalte]>]␣{Literal / Variable / Feld / Befehlsergänzung}␣

[[<LOC␣Zeile[,Spalte]>]␣{Literal / Variable / Feld / Befehlsergänzung}]...

Funktion:

Der SHOW-Befehl zeigt eine Nachricht auf dem Bildschirm an. Die Angabe von Zeile und Spalte legt die Position der Nachricht auf dem Bildschirm fest. Zeile und Spalte können als numerische Literale oder Variable codiert werden. Die Angabe von Variablen erlaubt das Anzeigen von Nachrichten auf verschiedene Bildschirmpositionen durch einen Befehl (z.B. in einem Unterprogramm).

Die Angabe [<LOC␣Zeile[,Spalte]>] kann mehrfach in einem Befehl auftreten. Dabei braucht keine bestimmte Reihenfolge der Bildschirmpositionen eingehalten werden. Bei Festlegung der Bildschirmposition ist zu beachten, daß Zeile 1 die Statuszeile und Zeile 2 die Fehlerzeile ist. Beide Zeilen werden ggfs. überschrieben (Vorsicht!).

Wird keine Positionsangabe codiert, wird die Nachricht auf dem 480-Zeichen Bildschirm in der Fehlerzeile (Zeile 2) und auf dem 1920-Zeichen Bildschirm in Zeile 1 ab Position 41 angezeigt.

Die Nachricht kann aus einer beliebigen Kombination von Literalen, Variablen und Feldern bestehen. Ein Teil der im Abschnitt 5.6.26 beschriebenen Befehlsergänzungen können ebenfalls verwendet werden.

	Befehlsstruktur	SHOW

Als Befehlsergänzungen sind <CRT> , <BATCH> , <JOB> , <DATE␣x> , <ALL> und @ALL@ zulässig. Die Anweisungen <TOP> und <LF> haben Sonderfunktionen. SHOW <TOP> löscht den Bildschirm ab Zeile 3; SHOW <LF> bewirkt einen Roll-Up um eine Zeile.

Die Befehlsergänzung <CRT␣480> setzt die Bildschirmgröße auf 480 Zeichen, <CRT␣1920> setzt die Bildschirmgröße auf 1920 Zeichen (nur möglich, wenn ein Bildschirmplatz Modell 3 mit 1920 Zeichen Kapazität benutzt wird).
Die Befehlsergänzung <CRT> ist in allen Editor-Programmen zulässig.

Felder und Variable können mit Maskenangaben (siehe Punkt 5.6.25.1 - 5.6.25.7) versehen werden. Ein Literal innerhalb des SHOW-Befehls darf max. 40 Stellen lang sein. Soll ein längeres Literal angezeigt werden, muß es in mehrere Literale unterteilt werden.

Eine Nachricht, die aus mehr als 200 Zeichen besteht, erzeugt einen Fehler zur Laufzeit des Programms (Vorsicht!).

Im Oberprüf-Modus und in der ABL-Funktion wird der SHOW-Befehl ignoriert.

Folgt eine SHOW-Anzeige einer anderen, werden nur soviele Positionen der alten Anzeige überschrieben, wie die neue Anzeige stellenmäßig benötigt. Das bedeutet: Folgt eine kurze Anzeige einer langen Anzeige, wird die kurze Anzeige zusammen mit dem Ende der langen Anzeige eingeblendet. Daher ist es empfehlenswert, im SHOW-Befehl die für die Anzeige nicht benötigten Positionen mit Blanks aufzufüllen.

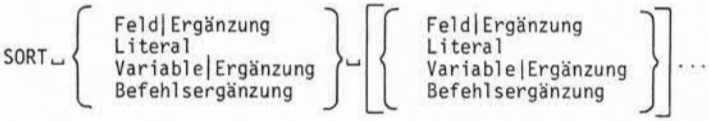
Beispiel:

```
SHOW␣<LOC␣4,10>␣'BITTE␣DATUM␣EINGEBEN:'.
ACCEPT␣<LOC␣4,32>␣DATUM.
```

	Befehlsstruktur	SORT

5.6.35 SORT

Format:



Funktion:

Der SORT-Befehl verkettet die Sätzes eines Stapels anhand der angegebenen Sortierbegriffe in auf- oder absteigender Reihenfolge.

Der Sortierschlüssel wird in der Reihenfolge der nach SORT angegebenen Operanden gebildet. Wird ein Feld oder eine Variable als Operand verwendet, kann durch die Ergänzung |AK (aufsteigende Folge) oder |DK (absteigende Folge) die Sortierfolge der einzelnen Operanden bestimmt werden. Ohne die Ergänzung |AK oder |DK wird aufsteigende Reihenfolge angenommen.

Durch eine Positionsangabe im SORT-Befehl, die unmittelbar nach dem Operanden bzw. nach |AK oder |DK gemacht wird, ist es möglich, bei Gleichheit von Sortierbegriffen innerhalb eines Stapels für den Sortier-Stapel eine bestimmte Reihenfolge vorzugeben.

Die Positionsangabe wird in Hochkommas (' ') eingeschlossen.

Achtung! Eine ausführliche Beschreibung des Sortierprogramms finden Sie im Abschnitt 3.5

5.6.36 STOP

Format:

STOP.

Funktion:

Der STOP-Befehl bewirkt die Beendigung eines Editor-Programms und sperrt das Editor-Programm für die in der Verarbeitung befindlichen Stapel.

Im Feldende-Editor kehrt das Programm zur Eingabe zurück. Bei den nächsten Eingaben wird das Editor-Programm nicht mehr angesprochen.

Im Stapelende-Editor, Ausgabe- und Sortierprogramm wird das Programm abgeschlossen und nicht mehr durchlaufen.

Bemerkung:

Wird in einem Programm kein STOP-Befehl gegeben, hält das Programm nach dem letzten verarbeiteten Satz von selbst an, d.h. ein STOP-Befehl ist nur sinnvoll, wenn ein vorzeitiger Abbruch des Programms gewünscht wird.

Der STOP-Befehl kann z.B. benutzt werden, um eine Enderoutine nach einer bestimmten Anzahl von Prüfungen zu verlassen.

Beispiel:

```
DECLARE␣A␣B.
WHEN␣FMT␣1␣GOTO␣!P1.
WHEN␣FMT␣2␣GOTO␣!P2.
RELEASE.
```

Achtung:

Wird im Sortierprogramm ein STOP-Befehl gegeben, bricht der Sort nach der 1. Phase (Aufbau der Arbeitsdatei mit Sortierschlüsseln und Adressen) ab. Die Angabe eines STOP-Befehls im Sortierprogramm ist daher unsinnig.

	Befehlsstruktur	STOP

```
!P1␣MOVE␣(1)␣TO␣A.
     MOVE␣(2)␣TO␣B.
RELEASE.

!P2␣IF␣A␣=␣0␣OR␣IF␣B␣=␣0␣GOTO␣!FEHL.
     OUTPUT ...
     .
     .
     .
     RELEASE.

!FEHL␣PAUSE␣'STAMMDATEN␣FEHLEN'.
     STOP.
```

In einem Ausgabeprogramm wird geprüft, ob wichtige Stammdaten erfaßt worden sind. Diese Daten (FMT 1) werden in Variable gebracht, um von einem anderen FMT (FMT 2) darauf zugreifen zu können. Sind diese Daten nicht erfaßt worden, oder wurde dieser Satz gelöscht, so läuft das System auf eine Fehlermeldung und anschließend auf einen STOP-Befehl.

Mit diesem STOP-Befehl wird der Editor sofort verlassen, da eine Ausgabe in diesem Fall sinnlos wäre.

5.6.37 SUBTRACT

Format:

SUBTRACT␣ { Literal / Feld / Variable / Arithmetischer Ausdruck } ␣FROM␣Variable.

Funktion:

Der SUBTRACT-Befehl subtrahiert den ersten Operanden vom zweiten Operanden. Das Ergebnis wird rechtsbündig im zweiten Operanden angestellt. Der erste Operand bleibt unverändert.

Die Länge des zweiten Operanden bleibt erhalten. Ist das Ergebnis länger als der zweite Operand, findet ein Überlauf statt und der Überlauf-Merker wird gesetzt. Der Merker kann vom Programm abgefragt werden. Der Überlauf geht verloren.

Wird ein alphanumerischer Operand verwendet, rechnet das System mit den rechten 4 Bit des EBCDIC-Codes.

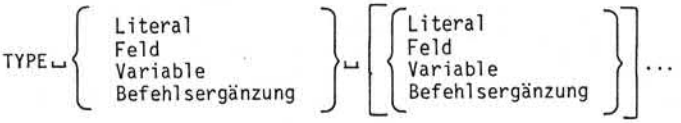
Beispiel:

```
(2)      = |0|0|0|2|4|
BESTAND  = |0|1|9|8|2|
SUBTRACT␣(2)␣FROM␣BESTAND.
BESTAND  = |0|1|9|5|8|
```

	Befehlsstruktur	TYPE

5.6.38 TYPE

Format:



Funktion:

Der TYPE-Befehl dient zur Datenausgabe auf dem Arbeitsplatzdrucker. Der TYPE-Befehl kann im Gegensatz zum OUTPUT-Befehl in allen Programmtypen verwendet werden.

Soll ein Arbeitsplatzdrucker angesprochen werden, der nicht an den eigenen Arbeitsplatz angeschlossen ist, wird durch die Befehlsergänzung <LOC␣n> der Arbeitsplatz definiert, auf dessen Drucker die Daten ausgegeben werden.

Beispiel:

```
TYPE␣<LOC␣3>␣ (2).
```

Der Inhalt von Feld 2 wird auf dem Drucker von Arbeitsplatz 3 ausgegeben.

Für weitere Funktionen des TYPE-Befehls gilt die Beschreibung des OUTPUT-Befehls.

Im Überprüf-Modus und in der ABL-Funktion wird der TYPE-Befehl ignoriert.

Die Blatthöhe des Arbeitsplatzdruckers wird durch die Befehlsergänzung <PAGE nn> angegeben (siehe Punkt 5.6.26.15).

Wird in einem TYPE-Befehl die Befehlsergänzung <DEFER> verwendet, beginnt die Befehlsergänzung <SKIP> im nächsten TYPE-Befehl ab der letzten Druckposition neu zu zählen.

5.6.39 WHEN

Format:

WHEN␣Bedingung␣Befehl[;␣Befehl]...

Funktion:

Der WHEN-Befehl testet bestimmte Systembedingungen und führt den oder die zugehörigen Befehle aus, wenn die Bedingung erfüllt ist. Ist die Bedingung nicht erfüllt, werden alle Befehle bis zum nächsten Punkt übergangen und erst der auf den Punkt folgende Befehl ausgeführt.

In einem WHEN-Satz können mehrere, durch Semikolon (;) getrennte Befehle enthalten sein. Der letzte Befehl des WHEN-Satzes muß durch einen Punkt abgeschlossen werden. Innerhalb eines WHEN-Satzes dürfen keine weiteren Bedingungsabfragen wie IF, WHEN oder ELSE auftreten.

Folgende Bedingungsabfragen sind möglich:

5.6.39.1 WHEN␣[NOT␣]FMT␣n

Die Anweisung prüft, mit welchem Eingabeformat ein Satz erfaßt wurde und knüpft an die Erfüllung bzw. Nichterfüllung der Bedingung eine weitere Bearbeitung.

"n" gibt die Nummer des Eingabeformats an und darf 0-9 sein.

Beispiel:

```
WHEN␣FMT␣1␣GOTO␣!TRANSP.
```

Wenn der anstehende Satz mit dem Eingabeformat 1 erfaßt wurde, verzweigt das System zur zum Label TRANSP.

```
WHEN␣NOT␣FMT␣3␣RELEASE.
ADD ...
```

Nur Sätze, die mit FMT 3 erfaßt wurde, werden selektiert und bearbeitet. Alle Sätze, die nicht mit FMT 3 erfaßt wurden, werden übergangen.

5.6.39.2 WHEN␣[NOT␣]@FMT␣n

Im Unterschied zur Funktion WHEN␣[NOT␣]FMT␣n wird das Format des Satzes in einer Indexdatei geprüft. Alle anderen Funktionen sind identisch.

5.6.39.3 WHEN␣[NOT␣]FIELD␣n

Diese Anweisung fragt die Nummer des zuletzt bearbeiteten Eingabefeldes ab. Die Ausführung des angegebenen Befehls ist abhängig von der Obereinstimmung bzw. Nichtübereinstimmung der abgefragten mit der in der Anweisung WHEN␣[NOT␣]FIELD angegebenen Feld-Nummer.

"n" repräsentiert die Feld-Nummer und kann eine Zahl zwischen 1 und 999 sein. Die Feld-Nummer wird nicht in Klammern eingeschlossen.

Die Anweisung ist nur im Feldende-Editor zulässig.

5.6.39.4 WHEN␣[NOT␣]ENTRY

Die Bedingung ist erfüllt (bzw. nicht erfüllt), wenn sich das System im Eingabe-Modus befindet.

5.6.39.5 WHEN␣[NOT␣]VERIFY

Die Bedingung ist erfüllt (bzw. nicht erfüllt), wenn sich das System im Prüf-Modus befindet.

Beachten Sie bitte, daß der Feldende-Editor im Prüf-Modus nur nach der Korrektur eines Feldes durchlaufen wird.

5.6.39.6 WHEN␣[NOT␣]UPDATE

Die Bedingung ist erfüllt (bzw. nicht erfüllt), wenn sich das System im UPDATE-Modus befindet.

5.6.39.7 WHEN␣[NOT␣]EXAMINE

Die Bedingung ist erfüllt (bzw. nicht erfüllt), wenn sich das System im Untersuchungs-Modus befindet.

Beachten Sie bitte, daß der Feldende-Editor im Untersuchungs-Modus nur nach der Korrektur eines Feldes durchlaufen wird.

5.6.39.8 WHEN␣START

Die Anlage führt den in der Anweisung WHEN START angegebenen Befehl aus, wenn der erste Satz des ersten Stapels verarbeitet wird.

- Wird nur ein Stapel verarbeitet, sind beim ersten Satz die Bedingungen WHEN␣START und WHEN␣FILE erfüllt.
- Werden mehrere Stapel bearbeitet, ist die Bedingung WHEN␣START nur beim ersten Satz des ersten Stapels erfüllt, während die Bedingung WHEN␣FILE jeweils beim ersten Satz der einzelnen Stapel erfüllt ist.

Beispiel:

```
WHEN␣START␣GOTO␣!KENNSATZ.
```

Wird er erste Satz des ersten Stapels abgearbeitet, verzweigt das Programm zum Label !KENNSATZ.
Die Bedingung WHEN START ist auch beim Fortsetzen eines Stapels erfüllt.

	Befehlsstruktur	WHEN

5.6.39.9 WHEN␣FILE

Die Bedingung WHEN␣FILE ist erfüllt, wenn der erste Satz eines Stapels verarbeitet wird. Die Anwendung der WHEN␣FILE-Abfrage ist sinnvoll, wenn mehrere Stapel verarbeitet und jeweils bei Stapelbeginn bestimmte Routinen durchgeführt werden sollen.

Beispiel:

```
WHEN␣FILE␣OUTPUT␣<JOB>␣<BATCH>.
```

Bei der Ausgabe mehrerer Stapel wird jeweils bei Stapelbeginn der Standard-Job-Name und der Stapelname ausgegeben.

```
WHEN␣FILE␣PERFORM␣!NEUBLATT.
```

Das Programm verzweigt zum Label !NEUBLATT, falls ein neuer Stapel abgearbeitet wird.

5.6.39.10 WHEN␣FLAG

Das System prüft, ob an irgendeiner Stelle des aktuellen Satzes ein Fehlerkennzeichen vorhanden ist.

Beispiel:

```
WHEN␣FLAG␣GOTO␣!FEHLER.
```

Falls sich an einer Stelle des Satzes ein Fehlerkennzeichen befindet, verzweigt das Programm zum Label !FEHLER.

5.6.39.11 WHEN␣@FLAG

Das System prüft, ob sich an irgendeiner Stelle des aktuellen Satzes einer Indexdatei ein Fehlerkennzeichen befindet.

5.6.39.12 WHEN␣OVERFLOW

Diese Anweisung prüft, ob bei einer arithmetischen Operation ein Oberlauf stattgefunden hat.

Beispiel:

```
WHEN␣OVERFLOW␣PAUSE␣'FELD␣ZU␣KLEIN␣UEBERLAUF'.
```

Findet bei einer arithmetischen Operation ein Oberlauf statt, ertönt der Fehlerton und in der Fehlerzeile des Displays erscheint "FELD ZU KLEIN, OBERLAUF".
Die Bedingung OVERFLOW muß sofort nach der Ausführung der arithmetischen Operation abgefragt werden.

5.6.39.13 WHEN␣EOT

Die Bandmarke wird während der Ausgabe auf Band abgefragt.

Beispiel:

```
WHEN␣EOT␣PERFORM␣!NACHSATZ.
```

Wurde die Bandendemarke erreicht, verzweigt das Programm in das Unterprogramm NACHSATZ. Hier können z.B. Nachsätze und Bandmarken geschrieben und das Magnetband vom Programm zurückgespult werden.

```
WHEN␣EOT␣PAUSE␣'NEUES␣BAND␣EINLEGEN'.
```

Bei Erreichen der Bandendemarke erhält die Bedienungskraft die Meldung, einen Bandwechsel vorzunehmen.

Standard-Job

6. Standard-Job

Durch Anwahl eines Standard-Jobs werden eine Vielzahl von gespeicherten Parametern abgerufen, die andernfalls von der Bedienungskraft eingegeben werden müßten.

Bei folgenden Funktionen wird durch die Anwendung von Standard-Jobs die Bedienung erleichtert:

- Eröffnen eines Stapels

  Aus dem Standard-Job werden abgerufen:
  - Stapelnamenvorgabe
  - Namen der Eingabeformate und Verkettung
  - Name des Feldende-Editors
  - Name des Stapelende-Editors

- Dateibearbeitung

  Aus dem Standard-Job werden die gleichen Parameter wie beim Eröffnen eines Stapels abgerufen. Die Anwahl eines Standard-Jobs ist in der Funktion Dateibearbeitung zwingend vorgeschrieben.

- Ausgeben eines Stapels

  Aus dem Standard-Job werden abgerufen:
  - Ausgabeparameter
  - Ausgabeprogramm
  - Ausgabebedingungen

- Lesen eines Stapels

  Aus dem Standard-Job werden abgerufen:
  - Stapelnamenvorgabe
  - Namen der Eingabeformate und Verkettung

	Standard-Job

- Lesen mit RESCUE-Kennung

  Aus dem Standard-Job werden abgerufen:
  - Stapelnamenvorgabe
  - Namen der Eingabeformate und Verkettung

  Das Lesen mit RESCUE-Kennung ist nur bei Vorhandensein eines entsprechenden Standard-Jobs möglich.

Neben der erheblichen Bedienungsanleitung bietet der Standard-Job eine zusätzliche Sicherheit gegen Bedienungsfehler.

6.1 Standard-Job-Name

Die zu einem Standard-Job gehörenden Parameter werden auf dem Formular "Standard-Job" erfaßt und anschließend in das System eingegeben. Jeder Standard-Job wird unter einem max.8-stelligen Namen in der Standard-Job-Bibliothek gespeichert. Der Name kann aus Buchstaben oder Ziffern bestehen, jedoch muß das erste Zeichen zwingend ein Buchstabe sein.

Ein einmal gespeicherter Standard-Job kann unter seinem Namen beliebig oft aufgerufen werden.

6.2 Stapelnamenvorgabe

Im Feld Stapelnamenvorgabe kann eine Zeichenfolge festgelegt werden, die an dieser Stelle im Namen eines jeden Stapels enthalten sein muß, der unter diesem Standard-Job in das System eingegeben wird. Zeichenpositionen im Stapelnamen, die frei wählbare Zeichen enthalten können, werden in der Stapelnamenvorgabe durch Sternchen (*) gekennzeichnet.

Das Feld für Stapelnamenvorgabe ist 10-stellig und entspricht der Länge eines Stapelnamens.

	Standard-Job

Beispiel 1:

STANDARD-JOB-NAME	PERSONAL
STAPELNAMENVORGABE	PERS******

Alle Namen der Stapel, die über den Standard-Job PERSONAL erfaßt werden, müssen auf den ersten 4 Stellen die Zeichenfolge PERS haben. Die Sternchen-Vereinbarung besagt, daß für die mit "*" gekennzeichneten Stellen keine Bedingungen gesetzt werden.

Beispiel 2:

STANDARD-JOB-NAME	PERSONAL
STAPELNAMENVORGABE	***ABC***1

Der Stapelname muß auf den Stellen 4-6 die Zeichenfolge ABC und auf der letzten Stelle das Zeichen 1 aufweisen. Ist dies nicht der Fall, wird der Bedienungskraft der Fehler STAPELNAME FALSCH angezeigt. Sie kann den Stapel in diesem Fall nicht beginnen.

Soll ohne Stapelnamenvorgabe gearbeitet werden, wird auf allen Stellen "*" (Sternchen) angegeben.

STANDARD-JOB-NAME	PERSONAL
STAPELNAMENVORGABE	**********

Die Stapelnamen sind somit frei wählbar.

Die Stapelnamenvorgabe erleichtert wesentlich die weitere Bearbeitung von Stapeln, z.B. wenn mehrere Stapel, die inhaltlich zusammengehören, gleichzeitig verarbeitet werden sollen (Multi-Batch-Verarbeitung).

6.3 Eingabeformat Name

Die Namen der zu verwendenden Eingabeformate werden in diese Felder eingetragen.

	Standard-Job

6.4 Verweis auf

In einem Standard-Job können bis zu 10 Eingabeformate miteinander verkettet werden. Die Verkettung legt fest, in welcher Reihenfolge die einzelnen Eingabeformate abgearbeitet werden sollen. Die Eingabeformat-Namen werden unter der jeweiligen Nummer (Nummer 0 = Format 10) eingetragen. Unter "Verweis auf" wird die Nummer des in der Kette folgenden Eingabeformates angegeben.

Beispiel einer Formatverkettung:

Eingabeformate und Verkettung:

	Name	Verweis auf		Name	Verweis auf
1	K O P F	2	2	R E C H N Z E 1	3
3	R E C H N Z E 2	2	4	S U M M E	1
5			6		
7			8		
9			0		
Feldeditor			Stapeleditor		

Die Dateneingabe beginnt mit dem Eingabeformat K O P F. Anschliessend wird das Format R E C H N Z E 1 und R E C H N Z E 2 abgearbeitet. Danach verzweigt das System durch die Verkettung wieder in das Format 2. Das Format 4 S U M M E wird durch manuelle Formatwahl oder durch ein Kettfeld im Eingabeformat aufgerufen. Nach Abarbeitung des Formats S U M M E wird zum Format 1 K O P F verzweigt

	Standard-Job

6.5 Ausgabeparameter

Die folgenden Parameter steuern die Ausgabe eines Stapels nach Aufruf des Standard-Jobs:

6.5.1 Ausgabegerät

Als Ausgabegerät wird eine der 4 möglichen Magnetbandeinheiten, der Drucker oder die Datenfernübertragung (DFÜ) angegeben.

- T 1 (Tape 1) = Bandeinheit 1
- T 2 (Tape 2) = Bandeinheit 2
- T 3 (Tape 3) = Bandeinheit 3
- T 4 (Tape 4) = Bandeinheit 4
- P R (Printer) = Systemdrucker
- T C (Telecommunications) = Datenfernübertragung

Wird P R (Drucker) als Ausgabegerät angegeben, entfallen alle weiteren Angaben bis auf den Namen des Ausgabeprogramms.

6.5.2 Satzlänge (nur bei Bandausgabe)

Der Parameter Satzlänge gibt die Anzahl der Zeichen im Ausgabesatz an. Die maximale Satzlänge beträgt 4.096 Zeichen. Die Ausgabesätze können sowohl feste Länge (alle Sätze einer Ausgabedatei sind gleich lang) als auch variable Länge haben (die Sätze einer Ausgabedatei können unterschiedlich lang sein).

Bei fester Satzlänge wird die Anzahl der Zeichen im Ausgabesatz angegeben.

Bei variabler Satzlänge wird die Anzahl der Zeichen im längsten Ausgabesatz angegeben.

Wird keine Angabe über die Satzlänge gemacht, werden die Sätze in der Länge ausgegeben, wie sie das Ausgabeprogramm aufbereitet.

	Standard-Job

6.5.3 Blocklänge (nur bei Bandausgabe)

Bei der Bandausgabe ist es erheblich effizienter, mehrere Sätze zu einem Block zusammenzufassen, da das Schreiben eines Blockzwischenraums erst nach mehreren Ausgabesätzen erforderlich wird. Es ist möglich, sowohl Sätze fester Länge, als auch Sätze variabler Länge zu blocken.

- Bei Sätzen fester Länge muß die Blocklänge ein Vielfaches der Satzlänge sein.
- Bei Sätzen variabler Länge muß die Blocklänge ein Vielfaches der Länge des größten Satzes sein.

In jedem Fall beträgt die maximale Blocklänge 4.096 Zeichen.

Wird keine Angabe über die Blocklänge gemacht, nimmt das System bei geblockten Sätzen eine Blocklänge von 4.096 Zeichen an.

6.5.4 Feste Länge (nur bei Bandausgabe)

Dieser Parameter gibt an, ob mit fester oder variabler Satz- bzw. Blocklänge gearbeitet wird.

- Y (yes) Sätze (Blöcke) haben feste Länge
- N (no) Sätze (Blöcke) haben variable Länge

Wird keine Angabe gemacht, nimmt das System variable Länge an.

6.5.5 Geblockte Sätze (nur bei Bandausgabe)

Dieser Parameter legt fest, ob die Ausgabesätze geblockt werden.

- Y (yes) Die Sätze werden bei der Ausgabe geblockt. Das System füllt jeden Bandblock bis zur maximalen Länge mit ganzen Sätzen auf.

	Standard-Job

- N (no) Sätze werden nicht geblockt, d.h. jeder Ausgabesatz wird als separater Block ausgegeben.

- S Auffüllen einer vorgegebenen MB-Blocklänge mit Daten (spanned Data).
Mit der Parameterangabe "S" hat man die Möglichkeit, variable Satzlängen in festen MB-Blöcken ohne Füllzeichen auszugeben. (Der Vorteil kommt besonders bei Anwendung der OUTPUT-Befehlsergänzungen TS, TZ, LS und LZ zum Ausdruck).
Das bedeutet, daß mehrere logische MB-Sätze in einem MB-Block enthalten sein können und daß ein logischer Satz über zwei MB-Blöcke gehen kann.

  Beispiel:

  A A A A A A A A A A B B B B B MB-Block 1

  B B B B B C C ..... MB-Block 2

  A = Logischer Satz 1

  B = Logischer Satz 2

  C = Logischer Satz 3

- K Die Funktion "K" arbeitet analog zur Funktion "S", mit dem Unterschied, daß zusätzlich die erste Stelle eines jeden MB-Blockes mit einer Konstanten (in der Supervisor-Ebene im Hex-Code festlegbar) beschrieben wird.

Achtung! Wird S oder K angegeben, muß unter FESTE LÄNGE "N" eingetragen werden.

	Standard-Job

6.5.6 Zeichenzählung (nur bei Bandausgabe)

Die Angabe "Zeichenzählung" ist nur in Verbindung mit der Angabe "Variable Satz- und Blocklänge" wirksam. Das System ermittelt für jeden Ausgabeblock die Zeichenanzahl und stellt sie am Anfang des Ausgabeblockes ab. Bei der Angabe Satz-/Blocklänge braucht die Anzahl der Stellen, die für die Darstellung des Zeichenzählers beansprucht werden, nicht brücksichtigt werden.

Folgende Arten der Zeichenzählung sind möglich:

D = Dezimale Zählung

B = Binäre Zählung

S = Füllzeichen (siehe Parameter "Füllzeichen" im Standard-Job)

Die Zeichenzählung kann maximal 4-stellig erfolgen. B und D können nicht gemeinsam angegeben werden.

Beispiel:

DDDD Der Zähler wird auf 4 Dezimalstellen abgestellt.

BBSS Der Zähler wird auf 2 Binärstellen abgestellt. Zusätzlich werden hinter dem Zähler zwei Füllzeichen in den Ausgabesatz eingefügt.

6.5.7 Füllzeichen (nur bei Bandausgabe)

Werden geblockte Sätze mit fester Länge ausgegeben und der letzte Block nicht vollständig mit Daten gefüllt, muß der Block mit Füllzeichen aufgefüllt werden.

	Standard-Job

Werden variabel lange Sätze/Blöcke ausgegeben, fügt das System beim Erkennen eines "S" im Parameter "Zeichenzählung" oder in der Befehlsergänzung <COUNT> ein Füllzeichen in den Ausgabesatz ein.

Für 9-Kanal-Magnetbänder gibt es zwei mögliche Füllzeichenangaben:

- Hexadezimaler Code (2-stellig)
- Blank, gefolgt von dem gewünschten Tastaturzeichen

Bei 7-Kanal-Magnetbändern kann kein hexadezimaler Code angegeben werden.

6.5.8 Ausgabecode (nur bei Bandausgabe)

Hier wird der Name des gewünschten Ausgabecodes (Fremdcodes) angeben. Der Fremdcode wird über die Funktion "BIBLIOTHEKEN" (Supervisorübersicht) in das System eingegeben.

Wird im Feld "Ausgabecode" des Standard-Jobs keine Eintragung gemacht, verwendet das System den System-Standard-Ausgabecode.
Bei 7-Kanal-Magnetbandeinheit muß ein Ausgabecode angegeben werden.

6.5.9 Ausgabeprogramm

Hier wird der Name des Ausgabeprogramms angegeben.

	Standard-Job

6.6 Ausgabebedingungen

Die Ausgabebedingungen verhindern das unerwünschte Ausgeben eines Stapels. So kann z.B. festgelegt werden, daß nur vollständig geprüfte Stapel oder Stapel, die keine Stapelsummendifferenz aufweisen, ausgegeben werden.

Die Ausgabebedingungen sind in Form einer Tabelle aufgebaut, wobei jede Bedingung, die zur Ausgabe eines Stapels erfüllt sein muß, mit einem "X" gekennzeichnet wird.

	Standard-Job

6.7 Codierformular

WIRD NACHGETRAGEN

Funktionsfolgen/Job-Control-Sprache

7.1 Funktionsfolge

Die Funktionsfolge ist ein programmierter Ablauf von Funktionen, d.h. eine vom System simulierte Anwahl von Funktionen und Tastenanschlägen, die normalerweise der Bediener vornimmt.

Die Anwendung der Funktionsfolge entlastet den Bediener bei ständig wiederkehrenden Routinen und gewährleistet, daß Fehler vermieden und die Reihenfolge der Operationen eingehalten wird.

Die Funktionsfolge wird über die Tastatur in der Funktionsfolge-Bibliothek im 3-Zeichen-Format eingegeben, auf der Magnetplatte abgespeichert und nach Aufruf automatisch abgearbeitet.

Typische Beispiele für die Anwendung einer Funktionsfolge sind:

- Ausgabe auf Band mit Standard-Kennsätzen.
- Sortieren, Ausgeben und Initialisieren von Bedienerstatistiken.
- Setzen von Systembedingungen bei der ABL-Funktion, z.B. Tabellen zuweisen, Stapelschutzbedingungen setzen.
- Bedienerlose Datenübertragung.

7.1.1 Elemente der Funktionsfolge

Jede Funktionsfolge besteht aus zwei unterschiedlichen Elementen: Operationscodes (op-code) und vom Benutzer frei wählbaren Tastaturzeichen.

7.1.1.1 Operationscode

Operationscodes bestehen aus zwei Zeichen und repräsentieren mit wenigen Ausnahmen Tastenfunktionen.

Die Ausnahmen sind:

- KY Erlaubt die Eingabe von Tastaturzeichen (z.B. zur Eingabe von Programmnamen, Stapelnamen etc.)

  Tritt dieser Befehl in der Funktionsfolge auf, wartet das System und akzeptiert alle Tastenanschläge der Tastatur. Sobald die Tastenkombination "FMT" und "AUSL" gleichzeitig betätigt wird, setzt das System die Abarbeitung der Funktionsfolge fort und zwar mit dem Befehl, der innerhalb der Funktionsfolge unmittelbar nach dem KY-Befehl steht.

- ST Start Einzelschritt

- SP Stop Einzelschritt

  Die Anweisung ST gibt an, daß bei Abarbeitung einer Funktionsfolge im Einzelschritt gearbeitet werden soll, d.h. jeder OP-Code/jedes Zeichen der Funktionsfolge wird ab dieser Position durch das Drücken der Cursor-rechts-Taste (→) ausgeführt. Es wird solange im Einzelschritt gearbeitet, bis

  - der Einzelschritt durch die Anweisung SP (Stop Einzelschritt) aufgehoben wird,
  - die Funktionsfolge beendet ist,
  - der Ablauf der Funktionsfolge durch FMT-DIAL unterbrochen wird.

  Die Möglichkeit, Funktionsfolgen im Einzelschritt ablaufen zu lassen, ist für Testzwecke interessant. Ist der Ablauf der Funktionsfolge nach der Testphase korrekt, kann die Anweisung ST bzw. SP durch "xx" (NOP-Befehle) ersetzt werden.

- xx No-Op

  Dieser Code wird vom System ignoriert.

In der folgenden Tabelle werden alle Op-Codes mit den dafür zulässigen Tastaturzeichen aufgeführt:

Op-Code	Beschreibung der Funktion	Zeichen
ØØ		alle Tastaturzeichen zulässig
NS	Negatives Oversign	0 - 9
PS	Positives Oversign	0 - 9
CL	Cursor nach links (rückwärts)	Ø = 1 Zeichen F = 1 Feld R = 1 Satz
CR	Cursor nach rechts (vorwärts)	Ø = 1 Zeichen F = 1 Feld R = 1 Satz
CO	Korrigieren	Ø = 1 Zeichen F = 1 Feld R = 1 Satz
DL	Löschen	Ø = 1 Zeichen R = 1 Satz
IN	Einfügen	Ø = 1 Zeichen R = 1 Satz
FM	Formatwahl	0 - 9
RC	Satz	D = Masken anzeigen/ausbl. E = Setzen Fehlerkennzeichen auf letzte Stelle F = Setzen Fehlerkennzeichen R = Suchen wiederholen S = Fehlerkennzeichen suchen T = Sprung auf nächstes Tab.Feld
AU	Auto-Taste (AUTO)	Zeichenangabe entfällt
DP	Duplizieren (DUP)	Zeichenangabe entfällt
FR	Feld-Auslösetaste (FELD-AUSL)	Zeichenangabe entfällt
HL	Dialog-Taste (DIAL)	Zeichenangabe entfällt
KY	Akzeptiert Tastenanschl.	Zeichenangabe entfällt

Op-Code	Beschreibung der Funktion	Zeichen
LR	Location Return (↵)	Zeichenangabe entfällt
RL	Auslösetaste (AUSL)	Zeichenangabe entfällt
RS	Reset-Taste (RESET)	Zeichenangabe entfällt
**	Keine Bedeutung (NOP-Befehl)	Zeichenangabe entfällt
ST	Start Einzelschritt	Zeichenangabe entfällt
SP	Stop Einzelschritt	Zeichenangabe entfällt

7.1.1.2 Tastaturzeichen

Treten Tastaturzeichen in Verbindung mit Op-Codes auf, dienen sie zur Modifizierung der Funktion (siehe Tabelle der Op-Codes).

Tastaturzeichen ohne Op-Code gelten als Eingabezeichen und können z.B. Programmnamen, Stapelnamen, Anwahl von Funktionen aus Funktionswahltabellen etc. darstellen.

7.1.2 Codierung der Funktionsfolge

Die Funktionsfolge erhält einen Namen, der maximal 8 Stellen lang sein darf. Das 1.Zeichen des Namens muß ein Buchstabe sein. Der Rest des Namens kann aus Buchstaben oder Ziffern bestehen.

Die Funktionsfolge wird auf dem Formular "Funktionsfolge" im 3-Zeichen-Format vorgenommen.

Jeder Funktionsschritt besteht aus dem Op-Code und/oder einem Tastaturzeichen.

Funktionsfolgen/Job-Control-Sprache

7.1.3 Ausführen der Funktionsfolge

Nach der Anwahl der Funktion "FUNKTIONSFOLGE AUSFUEHREN" aus der Supervisor-Funktionsübersicht beginnt die Abarbeitung der Funktionsfolge.

Die Bildschirminhalte wechseln genauso wie nach einer manuellen Anwahl der einzelnen Funktionen mit dem Unterschied, daß durch die hohe Geschwindigkeit bei der Simulation der Tastenanschläge und Eingaben ein Verfolgen der Einzelschritte kaum möglich ist.

Alles läuft automatisch ab, bis entweder

- das Ende der Funktionsfolge erreicht wird,
- der Befehl KY (Akzeptiere Tastenanschläge) auftritt,
- eine nicht programmierte Fehlermeldung erscheint,
- ein manueller Stop durch FMT-DIAL gegeben wird.

Bemerkung:

Es besteht die Möglichkeit, durch eine Funktionsfolge eine andere Funktionsfolge aufzurufen, vorausgesetzt, der Aufruf erfolgt als letzte Anweisung in der Funktionsfolge.

7.1.4 Ausführen im Einzelschritt

Eine Funktionsfolge kann zu Testzwecken im Einzelschritt ausgeführt werden. Voraussetzung ist, daß zu Anfang der Funktionsfolge der Op-Code ST (Start Einzelschritt) eingesetzt wird. Ab dem Op-Code ST wird jeder Funktionsschritt erst nach Drücken der Cursor-rechts-Taste (→) ausgeführt.

Die Verarbeitung im Einzelschritt endet, wenn

- der Op-Code SP erreicht wird,
- die Funktionsfolge beendet ist,
- die Funktionsfolge durch FMT-DIAL abgebrochen wird.

7.1.5 Beenden der Funktionsfolge

Das Ende der Funktionsfolge ist erreicht, wenn das letzte Zeichen der Funktionsfolge abgearbeitet ist. Das System hält mit der Anzeige desjenigen Bildschirminhalts an, der durch die letzte Anweisung der Funktionsfolge erreicht wurde. Der Bediener kann normal weiterarbeiten.

Der Bediener kann jederzeit den Ablauf einer Funktionsfolge durch Betätigen der Tastenkombination FMT und DIAL abbrechen. Die Funktionsfolge wird unmittelbar nach Betätigen der Tastenkombination abgebrochen und die Kontrolle an das Terminal übergeben.

Wird FMT-DIAL während einer E/A-Operation betätigt, wird nur die Operation abgebrochen, die Funktionsfolge läuft weiter. Um die Funktionsfolge zu stoppen, muß erneut FMT-DIAL betätigt werden.

7.1.6 Fehlermeldungen

Treten während der Abarbeitung der Funktionsfolge Fehlermeldungen auf, prüft das System, ob der nächste Befehl innerhalb der Funktionsfolge eine RESET-Anweisung ist.

- Ist dieser Befehl ein RESET-Befehl, interpretiert das System den aufgetretenen Fehler als "programmierten Fehler" und fährt mit der Ausführung der Funktionsfolge fort (z.B. DATEI-ENDE beim Finden einer Bandmarke auf dem Magnetband).

- Ist dieser Befehl kein RESET-Befehl, sind hardwaremäßige Voraussetzungen nicht erfüllt, wie z.B. Magnetband nicht bereit, Drucker nicht eingeschaltet usw. Die Funktionsfolge wird unterbrochen und das Terminal in den KY (Akzeptiere Tastenanschläge)-Mode gesetzt.

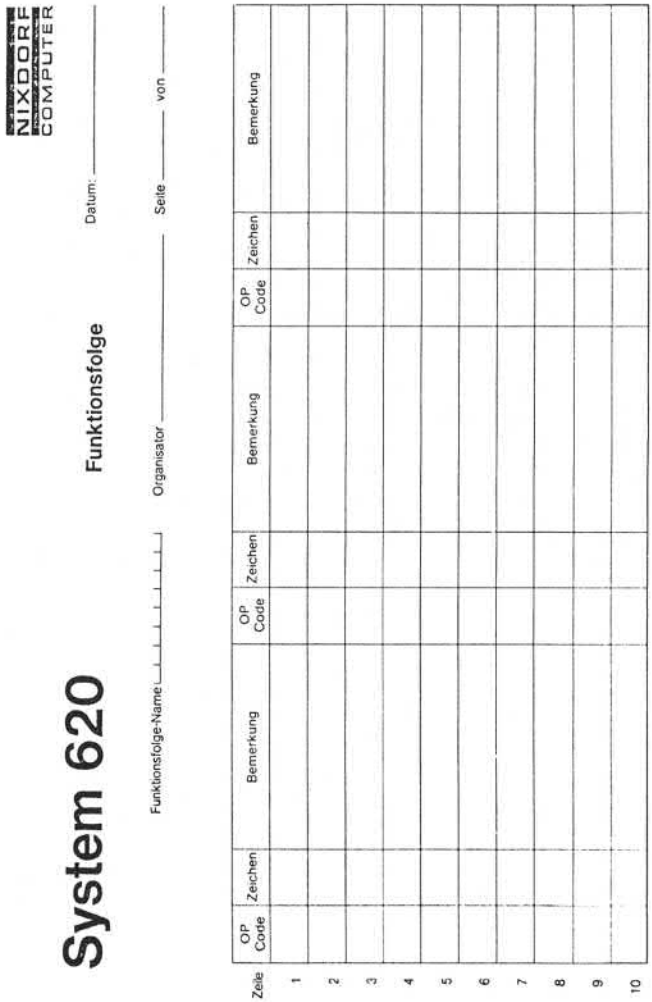
  Nun ist es Aufgabe des Bedieners, alle Schritte zu unternehmen, damit die Funktionsfolge weiter ablaufen kann, d.h. der Bediener muß z.B. die RESET-Taste betätigen und das Bandgerät betriebsbereit machen oder den Drucker einschalten.

  Bevor die Funktionsfolge fortgesetzt wird, muß die AUSL-Taste gedrückt werden. Sie ist in der Funktionsfolge bereits abgearbeitet und führte zur Hardware-Fehlermeldung. Wird nur die RESET-Taste gedrückt, ist der weitere Ablauf der Funktionsfolge falsch.

Anschließend wird die Tastenkombination FMT und AUSL betätigt. Die Funktionsfolge läuft weiter ab. Wird die Tastenkombination FMT und DIAL betätigt, wird die Ausführung der Funktionsfolge abgebrochen.

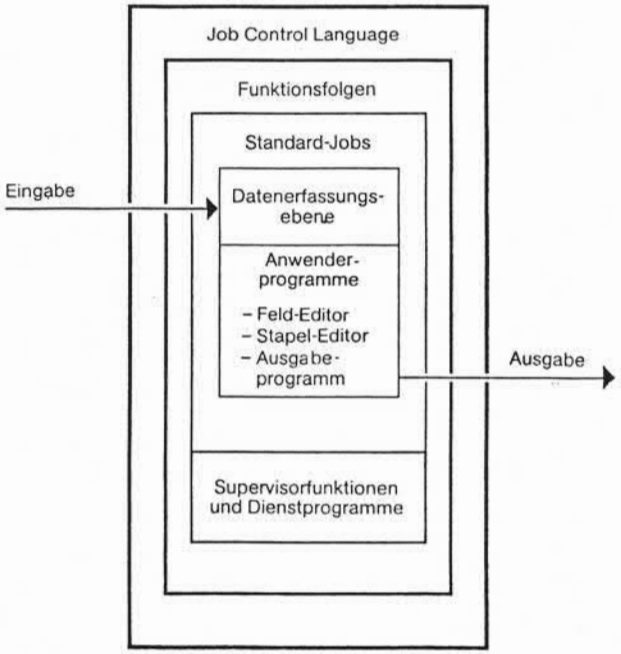
Wurde eine Funktionsfolge nicht ordnungsgemäß beendet, können bei Neustart Fehler auftreten. Der Grund liegt darin, daß nach Wiederanwahl am Unterbrechungspunkt fortgefahren wird. Durch Betätigen der Tastenkombination FMT-DIAL vor dem Neustart kann der Fehler umgangen werden.

7.1.7 Codierformular



7.2. Job Control Language (Systemsteuersprache)

Job-Control-Programme bilden eine Kontroll- und Steuerebene, die es erlaubt, Funktionsfolgen nach logischen Kriterien miteinander zu verknüpfen und das System ohne Eingriff des Bedieners zu steuern. Job-Control-Programme stehen an höchster Stelle der Software-Hierarchie des Systems 620. Grafisch kann diese Hierarchie wie folgt dargestellt werden:



Job-Control-Programme werden in der Supervisor-Ebene des Systems über die Funktion BIBLIOTHEKEN eingegeben und nach Aufruf über die Funktion JCL-PROGRAMM AUSFOHREN abgearbeitet. Jeder Bildschirmplatz kann mit einem eigenen Job-Control-Programm arbeiten. Die Anzahl der Job-Control-Programme ist nicht begrenzt.

7.2.1 Befehlstypen

Job-Control-Programme werden mit einer Untermenge der Editor-Befehle erstellt.
Insgesamt stehen 12 verschiedene Instruktionen zur Verfügung:

- ADD - Addition einer Variablen oder eines Literals zu einer Variablen.
- SUBTRACT - Subtraktion einer Variablen oder eines Literals von einer Variablen.
- MOVE - Übertragen einer Variablen oder eines Literals zu einer Variablen.
- IF - Logischer Vergleich zwischen einer Variablen und einem Literal oder zwischen zwei Variablen.
- GOTO - Unbedingte Verzweigung.
- PERFORM - Verzweigung in eine JCL-Unterroutine.
- SHOW - Anzeigen einer Variablen oder eines Literals.
- PAUSE - Anzeigen einer Variablen oder eines Literals mit Programmhalt und Fehleranzeige.
- ACCEPT - Dateneingabe über die Tastatur in eine Variable.
- NOTE - Kommentar einfügen.
- EXECUTE - Aufruf einer Funktionsfolge.
- WHEN - Abfrage, ob ein bestimmtes Peripheriegerät verfügbar ist.

7.2.2 Operanden

In Job-Control-Programmen sind folgende Operandentypen zulässig:

- Platzvariable
- Numerische Literale
- Alphanumerische Literale
- Befehlsergänzungen

7.2.2.1 Platzvariable

Für jeden Bildschirmplatz des Systems stehen 11 Platzvariable zur Verfügung. Diese Variablen brauchen nicht deklariert zu werden und werden auch nicht automatisch gelöscht.

Auf Platzvariable kann sowohl aus Editor-Programmen als auch aus Job-Control-Programmen zugegriffen werden. Ein Datenaustausch zwischen den unterschiedlichen Programmebenen und zwischen nacheinander ablaufenden Programmen ist somit möglich.

Für das Format der Platzvariablen gilt die Beschreibung unter Punkt 5.2.4 und 5.2.5. Die Namen der Platzvariablen müssen folgenden Aufbau entsprechen:

$VARnn

wobei nn ein Wert zwischen Ø1 und 11 sein kann.

7.2.2.2 Numerische Literale

Anstelle einer Variablen kann auch eine numerische Zeichenfolge unmittelbar als Operand eingesetzt werden.

Das numerische Literal ist maximal 14 Stellen lang und darf aus den Ziffern 0-9 bestehen.

Handelt es sich um einen negativen Wert, wird die Ziffernfolge mit einem Oversign versehen. Das Oversign kann an jeder Stelle innerhalb der Ziffernfolge auftreten, außer an der 1. Stelle.

7.2.2.3 Alphanumerisches Literal

Das alphanumerische Literal ist eine Zeichenfolge, die anstelle einer Variablen unmittelbar als Operand eingesetzt werden kann.

Das alphanumerische Literal ist maximal 120 Stellen lang und wird entweder in Hochkomma (') oder Anführungsstriche(") eingeschlossen. Innerhalb des Literals kann jedes beliebige Zeichen auftreten, ausgenommen die Zeichen, in die das Literal eingeschlossen ist.

Ein Literal, das aus mehreren identischen Zeichen besteht, kann in abgekürzter Schreibweise codiert werden. Zu diesem Zweck wird die Länge des Literals als Zahl und sofort anschließend das Zeichen selbst -in Hochkomma oder Anführungszeichen eingeschlossen- angegeben.

7.2.2.4 Befehlsergänzung

Die Befehlsergänzungen <DATE> und <TIME> können in den Befehlen MOVE, IF, SHOW und PAUSE als Operanden verwendet werden. Eine genaue Beschreibung der Befehlsergänzungen finden Sie unter Punkt 5.6.26.7 bzw. 5.6.26.20.

7.2.3 ACCEPT

Format:

```
ACCEPT␣[<LOC␣Zeile[,Spalte]>]␣$VARnn␣
       [ELSE␣Befehl[;␣Befehl]...].
```

Funktion:

Der ACCEPT-Befehl stoppt den Programmablauf und erlaubt die Eingabe von Daten in eine Platzvariable. Die Dateneingabe kann an beliebiger Stelle des Bildschirms erfolgen. Eine genaue Funktionsbeschreibung des ACCEPT-Befehls finden Sie unter Punkt 5.6.1.

7.2.4 ADD

Format:

```
ADD␣{Literal}␣TO␣$VARnn.
    {$VARnn}
```

Funktion:

Beide Operanden werden addiert. Das Ergebnis wird rechtsbündig im zweiten Operanden abgestellt. Der erste Operand bleibt unverändert.

Die Länge des zweiten Operanden bleibt erhalten. Ist das Ergebnis länger als der zweite Operand, findet ein Überlauf statt und der Überlauf geht verloren.

7.2.5 EXECUTE

Format:

EXECUTE␣Funktionsfolge-Name.

Funktion:

Der EXECUTE-Befehl ruft die angegebene Funktionsfolge zur Ausführung auf. Der Name der Funktionsfolge darf nur auf der 1.Stelle Sonderzeichen enthalten. Der Rest des Namens muß aus Buchstaben oder Ziffern bestehen. Innerhalb eines Unterprogramms ist der EXECUTE-Befehl nicht möglich und führt zu einem Fehler zur Compilierzeit des Programms.

7.2.6 GOTO

Format:

GOTO␣Label.

Funktion:

Durch den Befehl GOTO wird der lineare Programmablauf verlassen und das Programm verzweigt mit einem Vorwärts- oder Rückwärtssprung zum angegebenen Label. Der Label kann aus 2-9 Zeichen bestehen. Das erste Zeichen muß zwingend ein Ausrufungszeichen sein, danach folgt ein Buchstabe. Der Rest des Labels kann aus Buchstaben oder Ziffern bestehen.

Der im GOTO-Befehl codierte Label wird an der Stelle des Programms wiederholt, auf die verzweigt werden soll.

Der Label kann vor oder hinter dem GOTO-Befehl liegen, d.h. sowohl Vorwärts- als auch Rückwärtssprünge sind möglich.

Ein Label, der im Programm als Ansprungpunkt verwendet wird, darf nicht durch einen Punkt abgeschlossen werden.

Die Anzahl der Label in einem Programm ist nicht begrenzt.

Achtung! Der GOTO-Befehl ist nur für Sprünge innerhalb eines Hauptprogramms oder eines Unterprogramms zulässig. Für einen Wechsel zwischen Haupt- und Unterprogramm darf er nicht benutzt werden.

7.2.7 IF

Format:

```
IF␣{$VARnn      }␣{=}␣{$VARnn      }␣
   {Literal     }  {≠}  {Literal     }
   {Befehlsergänzung}  {<}  {Befehlsergänzung}
                       {>}
        Befehl[;␣Befehl]...
```

Funktion:

Der IF-Befehl vergleicht zwei Operanden und führt den folgenden Befehl aus, wenn die Vergleichsbedingung erfüllt ist.

Als Operanden im IF-Befehl sind nur Platzvariable, Literale und Befehlsergänzungen zulässig.

Eine genaue Beschreibung des IF-Befehls finden Sie unter Punkt 5.6.16.

7.2.8 MOVE

Format:

```
MOVE␣{$VARnn      }␣TO␣$VARnn.
     {Literal     }
     {Befehlsergänzung}
```

Funktion:

Der Inhalt des ersten Operanden wird in den zweiten Operanden übertragen. Der Inhalt des zweiten Operanden wird überschrieben; der erste Operand bleibt unverändert.

Der Typ und die Länge des zweiten Operanden werden ggf. durch den MOVE-Befehl verändert (siehe Abschnitt 5.2.4).

Soll eine Platzvariable eine bestimmte Länge erhalten, so kann dies durch den Transport eines alphanumerischen Literals erreicht werden, z.B. MOVE '␣␣␣␣␣' TO $VAR 01. Eine anschließende Addition von Null auf die Platzvariable wandelt diese in eine numerische Variable der festgelegten Stellenzahl um.

Diese Operation ist immer dann notwendig, wenn mit einem ACCEPT-Befehl Werte bestimmter Länge eingegeben werden sollen.

Die Obertragung kann numerisch oder alphanumerisch sein und ist vom Typ des ersten Operanden abhängig:

- Ist der erste Operand numerisch, erfolgt die Obertragung numerisch. Die Zeichen werden im zweiten Operanden rechtsbündig abgestellt. Nicht belegte Zeichenpositionen werden links mit Nullen aufgefüllt.
- Ist der erste Operand alphanumerisch, erfolgt die Obertragung alphanumerisch. Die Zeichen werden im zweiten Operanden linksbündig abgestellt. Nicht belegte Zeichenpositionen werden rechts mit Leerzeichen aufgefüllt.

Ist der erste Operand länger als der zweite Operand, werden die überschüssigen Zeichen links bzw. rechts abgeschnitten.

Eine Befehlsergänzung gilt immer als alphanumerischer Operand.

7.2.9 NOTE

Format:

NOTE ␣ Kommentar.

Funktion:

Der Note-Befehl erlaubt das Einfügen von Kommentaren in ein Job-Control-Programm.

Als Kommentar kann jede beliebige Zeichenfolge mit Ausnahme des Punktes codiert werden. Der Punkt beendet den Kommentar. Der NOTE-Befehl wird nicht compiliert und hat keinen Einfluß auf den Programmablauf.

7.2.10 PAUSE

Format:

```
PAUSE␣[<LOC␣Zeile[,Spalte]>]␣{$VARnn
                              Literal
                              Befehlsergänzung}␣
      [[<LOC␣Zeile[,Spalte]>]␣{$VARnn
                               Literal
                               Befehlsergänzung}] ...
```

Funktion:

Der PAUSE-Befehl stoppt den Programmablauf, zeigt eine Nachricht auf dem Bildschirm an und schaltet den Fehlerton ein. Nach Betätigung der RESET-Taste wird der Fehlerton ausgeschaltet und das Programm fortgesetzt.

Eine genaue Beschreibung des PAUSE-Befehls finden Sie unter Punkt 5.6.27. Im Gegensatz zu dieser Beschreibung sind als Operanden nur Platzvariable, Literale und die Befehlsergänzungen <DATE>, <TIME> und <CRT> zulässig.

7.2.11 PERFORM

Format:

```
PERFORM␣Label.
```

Funktion:

Durch den PERFORM-Befehl verzweigt das Hauptprogramm in ein Unterprogramm, das mit dem im PERFORM-Befehl angegebenen Label beginnt.

Der Label kann aus 2-9 Zeichen bestehen. Das erste Zeichen muß ein Ausrufungszeichen sein, danach folgt zwingend ein Buchstabe. Der Rest des Labels kann aus Buchstaben oder Ziffern bestehen. Nach Abarbeitung des Unterprogramms wird das Hauptprogramm mit dem Befehl fortgesetzt, der dem PERFORM-Befehl folgt.

Jedes Unterprogramm muß unmittelbar nach dem Label mit dem Wort ENTER beginnen. Der erste Befehl des Unterprogramms schließt sich ohne Punkt an.

Am Ende des Unterprogramms wird nach dem Punkt des letzten Befehls das Wort EXIT codiert. Dadurch wird das Unterprogramm verlassen und im Hauptprogramm weitergearbeitet.

In der Regel wird ein Unterprogramm verwendet, um mehrfach benötigte Programmfunktionen nur einmal codieren zu müssen.

In einem linearen Programmablauf ist es unzulässig, ein Unterprogramm ohne den PERFORM-Befehl anzusprechen.
Dementsprechend muß vor jedem Unterprogramm der Befehl GOTO oder EXIT codiert werden.

Die Befehle PERFORM und EXECUTE sind innerhalb eines Unterprogramms nicht zulässig.

Ein Unterprogramm darf nicht durch den Befehl GOTO verlassen werden.

7.2.12 SHOW

Format:

```
SHOW ␣[<LOC ␣ Zeile[,Spalte]>]␣{$VARnn
                               {Literal
                               {Befehlsergänzung}␣
      [[<LOC␣ Zeile[,Spalte]>]␣{$VARnn
                               {Literal
                               {Befehlsergänzung}] ...
```

Der SHOW-Befehl zeigt eine Nachricht auf dem Bildschirm an.

Eine genaue Beschreibung des SHOW Befehls finden Sie unter Punkt 5.6.34. Im Gegensatz zu dieser Beschreibung sind als Operanden nur Platzvariable, Literale und die Befehlsergänzungen <DATE>, <TIME> und <CRT> zulässig.

7.2.13 SUBTRACT

Format:

```
SUBTRACT␣{$VARnn}␣FROM ␣$VARnn.
         {Literal}
```

Funktion:

Der SUBTRACT-Befehl subtrahiert den ersten Operanden vom zweiten Operanden. Das Ergebnis wird rechtsbündig im zweiten Operanden abgestellt. Der erste Operand bleibt unverändert.

Die Länge des zweiten Operanden bleibt erhalten. Ist das Ergebnis länger als der zweite Operand, findet ein Überlauf statt. Der Überlauf geht verloren.

7.2.14 WHEN

Format:

```
WHEN ␣ { $TPn }
       { $CRD } ␣ [NOT] ␣ AVAILABLE ␣
       { $PRT }
       { $FPn }

        Befehl [; ␣ Befehl] ...
```

Funktion:

Durch den Befehl WHEN kann abgefragt werden, ob ein bestimmtes Peripheriegerät verfügbar (available) ist. Das Gerät ist nicht verfügbar, wenn es entweder nicht an das System angeschlossen ist oder von einem anderen Bildschirmplatz benutzt wird.

Folgende Geräte können abgefragt werden:

- $TPn (n = A-D) = Magnetband 1-4
- $CRD = Kartenleser
- $PRT = Systemdrucker
- $FPn (n = A-B) = Diskette 1-2

Durch den WHEN-Befehl können Programmabläufe -soweit organisatorisch sinnvoll- verändert werden, sobald ein Peripheriegerät nicht verfügbar ist.

7.2.15 Programmbeispiel

Das folgende Programmbeispiel (JCL-Programm zur Steuerung der Datenfernübertragung) zeigt einige Möglichkeiten der Job-Control-Language.

```
                                    . BEISPIEL JCL-PROGRAMM
                                    . ================================================================================
                                    .
PERFORM !ENTSCH.                    . SPRUNG ZUM UNTERPROGRAMM MIT AUSWAHL-MOEGLICHKEIT
MOVE 0 TO $VAR01.                   . SETZEN DER $VAR 1 AUF 14 STELLEN
MOVE $VAR02 TO $VAR01:14.           . INHALT VON $VAR 2 AUF DIE STELLE 14 DER $VAR 1
MOVE 103 TO $VAR01:11-13.           . MERKER FUER DIE NOTWENDIGEN AUSGABE-PROGRAMME
IF $VAR02 = 2 GOTO !EMPFANG.        . ABFRAGE WELCHE OPERATION LAUFEN SOLL
EXECUTE JCL/PARA.                   . AUFRUF DER FUNKTIONSFOLGE -- JCL/PARA
!SENDEN                             .
IF $VAR03 = 2 EXECUTE JCL/SEPL.     . AUFRUFEN DER FUNKTIONSFOLGE -- JCL/SEPL
IF $VAR03 = 1 EXECUTE JCL/SEMB.     . AUFRUFEN DER FUNKTIONSFOLGE -- JCL/SEMB.
IF $VAR01:14 = 1 GOTO !SENDEN.      . ABFRAGE, OB SENDEN WIEDERHOLT WERDEN SOLL
!DRUCK                              .
WHEN STPA NOT AVAILABLE SHOW <LOC 3,10> . ABFRAGE, OB BANDSTATION "A" BEREIT IST
'MAGNETBAND NICHT BEREIT'; GOTO !DRUCK. . MELDUNG: BAND NICHT BEREIT; SHOW, DAMIT KEINE "RESET"-TASTE NOTWENDIG IST
IF $VAR02 = 1 GOTO !ENDE.           . ABFRAGE, OB OPERATION BEENDET WERDEN SOLL
EXECUTE JCL/DRUC.                   . AUFRUF DER FUNKTIONSFOLGE -- JCL/DRUC
MOVE 2 TO $VAR01:14.                . SETZEN EINES MERKERS.
MOVE 103 TO $VAR01:11-13.           . SETZEN MERKER FUER NACHFOLGENDE PROGRAMME
!EMPFANG                            .
IF $VAR03 = 1 EXECUTE JCL/EMMB.     . ABFRAGE, WELCHE FUNKTIONSFOLGE LAUFEN SOLL, BEI GLEICHHEIT WIRD FUFO -- JCL/EMMB
                                    . AUFGERUFEN.
IF $VAR03 = 2 EXECUTE JCL/EMPL.     . BEI GLEICHHEIT WIRD FUNKTIONSFOLGE -- JCL/EMPL AUFGERUFEN
IF $VAR01:14 = 2 GOTO !EMPFANG.     . ABFRAGE, WELCHE FUNKTION AUSGEFUEHRT WERDEN SOLL
IF $VAR03 = 2 EXECUTE JCL/MBOK.     . BEI GLEICHHEIT WIRD FUNKTIONSFOLGE -- JCL/MBOK -- AUFGERUFEN
IF $VAR03 = 2 GOTO !EMPFANG.        . ABFRAGE, WELCHE OPERATION LAUFEN SOLL
!ENDE                               .
IF $VAR03 ≠ 0 EXECUTE JCL/EMOK.     . SELEKTIERUNG VON FUNKTIONEN
IF $VAR01:14 = 2 EXECUTE JCL/STLO;  . GOTO !EMPFANG
MOVE 0 TO $VAR01:14.                . SETZEN MERKER
GOTO !ENCM.                         .
!ENTSCH ENTER SHOW <TOP>.           . UNTERPROGRAMM EINSPRUNG: LOESCHEN DES GANZEN BILDSCHIRMS
!SEEM                               .
SHOW <LOC 4,5> '         C M  T P   M D' . ANZEIGE AB ZEILE 4, SPALTE 5
     <LOC 5,5> 'SENDEN   1    1     2 ' .
     <LOC 6,5> 'EMPFANG  2    1     2 ' .
     <LOC 8,5> 'WAEHLE     '.           .
MOVE '0' TO $VAR02. MOVE '0' TO $VAR03. . SETZEN VON $VAR 2 UND $VAR 3 ZUR EINSTELLIGEN VARIABLEN
ADD 0 TO $VAR02. ADD 0 TO $VAR03.   . $VAR 2 UND 3 ZUM AKZEPTIEREN VON NUMERISCHEN WERTEN.
ACCEPT <LOC 8,15> $VAR02 ELSE PAUSE . EINGABE IN $VAR 2, BEI DIAL-TASTE FEHLERMELDUNG
'PROZEDUR EINGEBEN'; GOTO !SEEM.    .
IF $VAR02 < 1 OR > 2 PAUSE 'NUR - 1 - O . ABFRAGE NACH GUELTIGER EINGABE, SONST FEHLERMELDUNG UND RUECKSPRUNG ZUR EINGABE
DER - 2 - ZULAESSIG'; GOTO !SEEM.   .
!TPMD                               .
ACCEPT <LOC 8,22> $VAR03 ELSE PAUSE . EINGABE IN $VAR 3, BEI BETAETIGEN DER DIAL-TASTE FEHLERMELDUNG
'WAEHLE BAND ODER PLATTE'; GOTO !TPMD. .
IF $VAR03 < 1 OR > 2 PAUSE 'NUR - 1 - OD . ABFRAGE AUF GUELTIGE EINGABE; SONST FEHLERMELDUNG UND RUECKSPRUNG ZUR EINGABE
ER - 2 - ZULAESSIG'; GOTO !TPMD.    .
EXIT                                . UNTERPROGRAMM AUSGANG
!ENCM                               .
SHOW 'ENDE DER DATENFERNUEBERTRAGUNG'. . ANZEIGE, DASS DAS JCL-PROGRAMM BEENDET IST
```

Prüfziffernrechnung

8. Prüfziffernrechnung

In der Prüfziffern-Bibliothek können bis zu 15 verschiedene Prüfziffernverfahren gespeichert werden. Das Verfahren wird im Prüfziffern-Formular eingetragen und anschließend in das System eingegeben. In den folgenden Abschnitten werden die einzelnen Parameter des Prüfziffern-Verfahrens erklärt.

8.1 Modulo

Die Prüfziffer wird u.a. errechnet, indem durch den Modulus dividiert wird. Die Standard-Moduli sind 7,10 und 11.

8.2 Nicht komplementiert

- Die Angabe NICHT KOMPLEMENTIERT bedeutet, daß der Rest (R1), der sich bei der Division durch den Modulus ergibt, die Prüfziffer ist.
- Die Angabe KOMPLEMENTIERT bedeutet, daß der Rest (R1), der sich bei der Division durch den Modulus ergibt, vom Modulus subtrahiert wird. Das Ergebnis ist die Prüfziffer.

8.3 Konstanter Rest

Mit Hilfe des konstanten Restes kann die Sicherheit des Prüfziffernverfahrens erhöht werden.

Ist R1 gleich oder kleiner als der konstante Rest, wird R1 vom konstanten Rest subtrahiert. Ist R1 größer als der konstante Rest, wird der Modulus zum konstanten Rest addiert und R1 vom Ergebnis (Modulo + konstanter Rest) subtrahiert.

Eine mögliche Komplementierung erfolgt nach der Bearbeitung des konstanten Restes. Wird nicht mit KONSTANTEM REST gearbeitet, ist es unbedingt erforderlich, bei der Angabe KONSTANTER REST den verwendeten Modulus einzutragen.

	Prüfziffernrechnung

8.4 Typ

Es gibt 5 Methoden der Errechnung:

1. Gewichtete Summe der Zahlen:

   Jede Zahl der Basiszahl wird mit ihrer Gewichtung multipliziert. Die einzelnen Ziffern der Ergebnisse der Multiplikation werden addiert und die Summe dann durch den Modulus dividiert. Der Divisionsrest ist R1.

   Beispiel:

Basiszahl	9	3	5	4	2	PZ
Gewichtung	2	1	2	1	2	
	18	3	10	4	4	

   1 + 8 + 3 + 1 + 0 + 4 + 4 = 21 : 10 (Modulus) = 2 Rest 1 (R1)

   Der Rest = R1 ist die PZ bei nicht komplementierter Errechnung. Bei Komplementierung wird dann R1 vom konstanten Rest bzw. dem Modulus subtrahiert (10 - 1 = 9 = PZ).

2. Gewichtete Summe der Prdukte:

   Jede Zahl der Basiszahl wird mit ihrer Gewichtung multipliziert. Die einzelnen Ergebnisse der Multiplikation werden addiert und die Summe dann durch den Modulus dividiert. Der Divisionsrest ist R1.

   Beispiel:

Basiszahl	9	3	5	4	2	PZ
Gewichtung	2	1	2	1	2	
	18	3	10	4	4	

   18 + 3 + 10 + 4 + 4 = 39 : 10 (Modulus) = 3 Rest 9 (R1)

Prüfziffernrechnung

Bei nicht komplementierter Errechnung ist somit R1 gleich der Prüfziffer. Bei komplementierter Errechnung wird dann R1 (9) vom konstanten Rest oder dem Modulus (10) subtrahiert. Das Ergebnis der Subtraktion (10 - 9 = 1) ist dann die Prüfziffer.

3. Division der ganzen Zahl:

   Die ganze Zahl, einschließlich der Prüfziffer wird durch den Modulus dividiert, der Rest ist R1.

4. Division der Basis-Zahl:

   Die Basiszahl wird durch den Modulus dividiert, der Rest ist R1. Bei nicht komplementierter Errechnung wird R1 somit die Prüfziffer. Bei komplementierter Errechnung wird R1 vom Modulus subtrahiert, und das Ergebnis ist dann die Prüfziffer.

5. Geometrische Prüfziffernrechnung mit der Basis von 2er Potenzen:

   Diese Methode entspricht der Rechnungsart 1 (gewichtete Summe der Zahlen). Der Unterschied liegt in der Gewichtung. Die Ziffern der Basiszahl werden nicht unmittelbar mit den Zahlen der angegebenen Gewichtung multipliziert, sondern die Zahlen der Gewichtung legen fest, mit welcher Potenz der Zahl 2 die Basiszahl multipliziert werden soll. Die Potenzangabe kann zwischen 0 - 9 liegen.

   Beispiel:

Modulus	11
Basiszahl	631243
Gewichtung	0123

	Prüfziffernrechnung

Die Ziffern der Gewichtung sind wie folgt zu interpretieren:

0 entspricht 2^0
1 entspricht 2^1
2 entspricht 2^2
3 entspricht 2^3

Die Basiszahl wird -beginnend mit der letzten Stelle- mit der jeweiligen 2er Potenz multipliziert.

$$3 \times 2^0 + 4 \times 2^1 + 2 \times 2^2 + 1 \times 2^3 + 3 \times 2^0 + 6 \times 2^1$$

$$= 3 \times 1 + 4 \times 2 + 2 \times 4 + 1 \times 8 + 3 \times 1 + 6 \times 2$$

$$= 3 + 8 + 8 + 8 + 3 + 12$$

$$= 42$$

42 : 11 (Modulus) = 3 Rest 9

Bei nicht komplementierter Errechnung ist die Prüfziffer gleich 9. Bei komplementierter Errechnung ist die Prüfziffer gleich 2 (Modulus 11 minus Rest 9 = 2).

8.5 Ersetzen von

- Prüfziffer 10; der Anwender kann hier das Zeichen angeben, das bei einer Prüfziffer 10 eingesetzt werden soll.
- Prüfziffer 11; hier wird das Zeichen angegeben, das bei einer Prüfziffer 11 eingesetzt werden soll.

Bemerkung:

Wird ein Ersetzungszeichen angegeben, so muß unbedingt auch das zweite Zeichen angegeben werden. Wird kein Ersetzungszeichen angegeben und es tritt eine dieser Prüfziffern (10 oder 11) auf, so meldet das System Prüfziffernfehler.

Prüfziffernrechnung

- Fehlerkennzeichen; soll nicht das vom System zu setzende Fehlerkennzeichen als Fehlermerkmal benutzt werden, so kann der Anwender hier das Zeichen angeben, das das Systemfehlerkennzeichen ersetzen soll. Dies spezielle Fehlerkennzeichen gilt nur für die Prüfziffer. Die Systemfunktionen (Setze Fehlerkennzeichen, SATZ-E und SATZ-F) setzen auch weiterhin das Systemfehlerkennzeichen (#).

8.6 Gewichtung

Bis zu 20 Gewichtunsfaktoren könne für jede Prüfziffernrechnung definiert werden. Das Prüfziffernfeld oder auch mehrere aufeinanderfolgende Felder können jede Länge haben, nur die Gewichtung muß sich spätestens nach jeder 20.Stelle wiederholen. Bei der Eingabe der Gewichtung ist darauf zu achten, daß sich die erste Stelle der Gewichtung auf die letzte Stelle der Basiszahl, die zweite Stelle auf die vorletzte Stelle usw. bezieht. Sind weniger Gewichtungsfaktoren als Feldpositionen angegeben, so wiederholt sich die Gewichtung bis zur vollen Länge der Basiszahl.

Beispiel:

Eingegebene Gewichtung: 21

Basiszahl	1	2	3	4	5	6	7	8	9	0	2	4
Gewichtung	1	2	1	2	1	2	1	2	1	2	1	2

8.7 Beispiel

Nummer der Prüfziffer (1 - 15)	01
Modulo	11
Nicht komplementiert (Y)	_
Konstanter Rest	11
Typ (1 - 5)	2

1 Gewichtete Summe der Zahlen
2 Gewichtete Summe der Produkte
3 Division der ganzen Zahl
4 Division der Basis-Zahl
5 Geometrische Basis 2

	Prüfziffernrechnung

Ersetzen von

Prüfziffer 10 (Y)	Y
Zeichen	Ø
Prüfziffer 11 (Y)	Y
Zeichen	1
System Fehlerkennzeichen (Y)	
Zeichen	

Gewichtung - | 2 | 3 | 4 | 5 | 6 | 7 |

Feldinhalt 10 Stellen	3 7 4 5 9 0 3 2 0 7
Gewichtung	5 4 3 2 7 6 5 4 3 2
Multiplikations-Summe	15 + 28 + 12 + 10 + 63 + 15 + 8 + 0 + 14 = 165
Division	165 : 11 (Modulus) = 15 Rest 0
Komplementierung	11 (Konstanter Rest) - 0 = 11

Ersetzen von 11 mit 1.

Daher Prüfziffer = 1.

	Prüfziffernrechnung

8.8 Codierformular

System 620

Prüfziffer

Nummer der Prüfziffer (1-15) ⌊_⌋⌊_⌋

Modulo ⌊_⌋⌊_⌋

Nicht komplementiert (Y) ⌊_⌋

Konstanter Rest ⌊_⌋⌊_⌋

Typ (1-5) ⌊_⌋

1 Gewichtete Summe der Zahlen

2 Gewichtete Summe der Produkte

3 Division der ganzen Zahl

4 Division der Basis-Zahl

5 Geomtr. Basis 2

Ersetzen von

Prüfziffer 10 (Y) ⌊_⌋

Zeichen ⌊_⌋

Prüfziffer 11 (Y) ⌊_⌋

Zeichen ⌊_⌋

System Fehlerkennz. (Y) ⌊_⌋

Zeichen ⌊_⌋

Gewichtung ⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋⌊_⌋

Anhang

9. Anhang

9.1 Compiler-Fehlermeldungen

Wird nach der Eingabe eines Fehlerende-, Satzende-, Stapelende-Editors, Ausgabe- oder Sortierprogramms die Funktion "BEENDEN" angewählt, erscheinen - sofern das Programm Fehler enthält - Fehlermeldungen auf dem Bildschirm.

Eine Fehlermeldung setzt sich wie folgt zusammen:

- Die ersten drei Stellen geben die Seitenzahl der Bildschirmanzeige an.
- Die beiden mittleren Stellen geben die Zeilennummer innerhalb der Bildschirmanzeige an.
- Die letzten beiden Stellen geben den Fehlercode an.
- In der nächsten Bildschirmzeile wird die Fehlermeldung im Klartext angezeigt.

Reicht ein Bildschirminhalt zur Anzeige aller Fehlermeldungen nicht aus, werden nach Betätigung der AUSL-Taste weitere Fehlermeldungen angezeigt. Fehler werden immer mit der Nummer der Zeile angezeigt, in der sich der Fehler auswirkt.

Beispiel:

001 - 05 - 72

Diese Fehlermeldung besagt, daß in Zeile 5 der ersten Bildschirmanzeige eine Variable angesprochen wird, die noch nicht definiert wurde. Der Fehler liegt in diesem Fall nicht in Zeile 5, sondern ist auf den fehlenden DECLARE-Befehl zurückzuführen. Die Auswirkung dieses Fehlers wird in Zeile 5 festgestellt.

Fehler, die sich auf mehrere Programmstellen auswirken, werden entsprechend oft angezeigt. In dem o.g. Beispiel wird z.B. jedesmal, wenn die nicht definierte Variable angesprochen wird, ein Fehler gemeldet, obwohl die Ursache nur in der fehlenden Definition des DECLARE-Befehls liegt.

	Anhang

Fehler, die sich auf mehrere Programmstellen auswirken, werden entsprechend oft angezeigt. In dem o.g. Beispiel wird z.B. jedesmal, wenn die nicht definierte Variable angesprochen wird, ein Fehler gemeldet, obwohl die Ursache nur in der fehlenden Definition des DECLARE-Befehls liegt.

Liste der Compiler-Fehlermeldungen:

01 NUMERISCHES LITERAL NICHT MOEGLICH

02 ALPHA LITERAL NICHT MOEGLICH

03 FELDNUMMER NICHT MOEGLICH

04 VARIABLE NICHT MOEGLICH

05 ARITHMETISCHER AUSDRUCK NICHT MOEGLICH

06 MASKENANGABE NICHT MOEGLICH

Eine Maske ist nur bei OUTPUT, AUDIT, PAUSE, SHOW, SORT und TYPE zulässig.

07 BEFEHLSERGÄNZUNG NICHT MOEGLICH

Eine Befehlsergänzung ist nur bei OUTPUT, AUDIT, PAUSE, SHOW, SORT und TYPE zulässig.

08 NUMERISCHES LITERAL MAXIMAL 14 STELLEN

09 OPERANDENTYP KANN NICHT BESTIMMT WERDEN

10 ALPHA LITERAL FORMATFEHLER

Dem Eröffnungs-Hochkomma darf nicht sofort das Beendigungs-Hochkomma folgen. Mindestens ein Zeichen muß eingeschlossen sein.

11 ALPHA LITERAL MAXIMAL 120 STELLEN

Anhang

12 WIEDERHOLUNG MAXIMAL 1 ZEICHEN MOEGLICH

Wird bei einem alphanumerischen Literal ein Wiederholungsfaktor angegeben, darf nicht mehr als 1 Zeichen in Hochkomma bzw. Anführungsstriche eingeschlossen werden.

Eröffnungs- und Beendigungssymbol müssen identisch sein.

13 WIEDERHOLUNGSFAKTOR MAXIMAL 120

Format-Fehler bei alphanumerischen Literal.
Der Wiederholungsfaktor darf nicht größer als 120 sein.

14 VARIABLEN-/DATEINAME MAXIMAL 8 ZEICHEN

Name der Variablen zu groß.
Der Name einer Variablen darf nicht größer als 8 Stellen sein.

15 'EXIT' MUSS EINZIGE ANGABE SEIN

Falscher Gebrauch des EXIT-Befehls.
Die Angabe EXIT muß die erste und einzige innerhalb des Befehls sein.

16 FEHLER BEI ANWENDUNG 'DECLARE/DEFINE'

Die Angabe DECLARE/DEFINE muß die erste und einzige innerhalb des Befehls sein.

17 FELDSPEZ. FEHLER, FELD-NR. > 2047

Feldnummer muß numerisch und eine Zahl zwischen 1 und 2047 sein.

18 SYNTAXFEHLER ':', '@' ODER ')' ERWARTET

Einer Feldnummer darf nur ':', '@' oder ')' folgen.

19 FORMATFEHLER TEILFELD

Ein Teilfeld wird folgendermaßen angegeben:
Nummer des Anfangszeichens - Nummer des letzten Zeichens.
Die Nummer kann also 1 - 99 sein. Die Nummer des Anfangszeichens muß kleiner oder gleich der Nummer des letzten Zeichens sein.
Sind beide gleich, ist die Angabe des letzten Zeichens überflüssig. In diesem Fall muß auch der - (Bindestrich) nicht gesetzt werden.
Der Angabe eines Teilfeldes muß die Angabe ')' oder 'ə' folgen.

20 SPEZIFIKATIONSFEHLER MASKE

Die Maske und die Angaben PK oder SG können nicht zusammen einem Feld zugeordnet sein. Eine Maske darf nicht länger als 20 Zeichen sein.

21 MASKE KANN NICHT DEFINIERT WERDEN

22 VARIABLEN-/DATEINAME MAXIMAL 8 STELLEN

23 FEHLER IN BEFEHLSERGAENZ. <COUNT␣XXXX>

Die Befehlsergänzung <COUNT␣xxxx> benötigt eine weitere Angabe.

24 FEHLER IN BEFEHLSERGAENZ. <BLK␣N>

Die Befehlsergänzungen benötigen eine weitere Angabe. Diese Angabe darf maximal 5 sein.

25 FEHLER IN BEFEHLSERGAENZ. <HEX␣XX>

Die Befehlsergänzung <HEX␣xx> benötigt ein hexadezimales Zeichen.

Anhang

26 FEHLER IN BEFEHLSERGAENZUNG <ALL>

Die Benutzung einer weiteren Angabe bei der Befehlsergänzung ALL ist nicht zwingend. Wird eine weitere Angabe gemacht, darf sie nicht größer als die maximale Feldanzahl sein. Wird Anfangs- und Endfeld angegeben, muß zwischen beiden ein - (Bindestrich) gegeben werden, und die Endadresse muß größer sein als die Anfangsadresse.

27 ANGABE <SKIP> ODER <BSP> MAXIMAL 2047

28 BILDSCHIRM ZEILE MAX. 24, SPALTE MAX. 80

29 BEFEHLSERGAENZUNG MIT ">" ABSCHLIESSEN

30 BEFEHLSERGAENZUNG UNDEFINIERBAR

31 SEITENANGABE NUMERISCH VON 01-99

32 FEHLER IM BEFEHL 'DECLARE/DEFINE'

Die Namen der Variablen, die im DECLARE-Befehl angegeben werden, müssen durch Komma, Blank oder beides getrennt werden. Nach dem letzten Namen muß ein Punkt gesetzt werden. Das erste Zeichen einer Variablen muß ein Buchstabe (A-Z) oder Ziffern (0-9) sein.

33 LABEL MAXIMAL 8 STELLEN

Ein Label darf nicht größer als 8 Stellen (ausschließlich '!') sein. Das erste Zeichen muß ein Buchstabe sein und sich direkt an das Aufrufungszeichen anschließen.

34 NACH LABEL MUSS KOMMA ODER BLANK FOLGEN

Ein Label muß vom nachfolgenden Befehl durch Komma und/oder Blank getrennt sein.

35 BEFEHLSART KANN NICHT DEFINIERT WERDEN

36 FALSCHE ANGABEN ZW. OPERANDEN

Format-Fehler beim ADD-, SUBTRACT-, MULTIPLY-, DIVIDE- oder MOVE-Befehl.
Zwischen Quell- und Zielfeld muß entweder TO, FROM, TIMES oder INTO angegeben werden.

37 LABEL MUSS AN '!' ANSCHLIESSEN

Der Label muß sich unmittelbar an das Ausrufungszeichen anschließen.

38 FORMATFEHLER BEI 'GOTO' ODER PERFORM'

Ein Label im GOTO- oder PERFORM-Befehl darf nicht größer als 8 Stellen (ausschließlich Ausrufungszeichen) sein. Das erste Zeichen muß ein Buchstabe (A-Z) sein und sich unmittelbar dem Ausrufungszeichen anschließen.

39 FORMATFEHLER IM 'RELEASE' ODER 'BYPASS'

In Verbindung mit RELEASE und BYPASS ist nur AT END oder ELSE zulässig.

40 'WHEN' OPERAND UNDEFINIERBAR

Der Operand nach WHEN ist in diesem Programmtyp nicht zulässig.

41 TEILVARIABLE SPEZIFIKATION FALSCH

42 WHEN FMT NUMMER VON 0-9

43 FORMATFEHLER 'WHEN (NOT) FMT X' BLANK??

Alle Teile eines WHEN [NOT] Befehls müssen durch Blank getrennt werden.

44 ZWISCHEN FMT UND NUMMER MUSS BLANK SEIN

45 FORMATFEHLER IM 'IF' BEFEHL

Gültige IF-Formate siehe Abschnitt 5.6.16

Anhang

46 FORMATFEHLER IM SATZ - PUNKT FEHLT

47 NACH 'ENTER' DARF KEIN PUNKT STEHEN

48 PROGRAMM MUSS MIT PUNKT ABSCHLIESSEN

49 UNTERPROGRAMM MUSS MIT 'ENTER' BEGINNEN

50 VARIABLE/DATEI-NAME STELLE 1 ≠ A-Z

Das erste Zeichen eines Variablen- oder Dateinamens muß ein Buchstabe (A-Z) sein.

51 BEFEHL NICHT ERLAUBT

Der Befehl ist in diesem Programmtyp nicht erlaubt. Siehe Tabelle in Abschnitt 3.

52 OUTPUT IM FELDENDE-/SORTPROG UNZULAESSIG

53 MAXIM. ANZAHL VARIABLE UEBERSCHRITTEN

Maximal 11 Variable im Feldende-Editor.
Maximal 99 Variable in allen anderen Programmtypen.

54 OPERAND BEI 'PAUSE' GROESSER 40 STELLEN

Ein alphanumerisches Literal im PAUSE-Befehl darf nicht länger als 40 Stellen sein.

55 'AT END' NICHT IM FELDENDEEDITOR ERLAUBT

57 FALSCHE 'USING' ANGABE

Der Operand nach USING muß ein Literal, ein Feld oder eine Variable sein. Teilfelder oder Teilvariable sind nicht zulässig.

58 FORMATFEHLER 'USING' ODER 'ELSE'

59 DATEI NAME UNDEFINIERBAR

Dateiname ist nicht durch DEFINE definiert.

	Anhang

60 MODIFIKATIONSFEHLER BEI DATEI

Im OPEN-Befehl ist nur UPD oder INC zulässig.

61 MAXIMAL 64 DATEIEN MOEGLICH

Mehr als 64 Dateien im DEFINE-Befehl angegeben.

62 FEHLER 'ELSE INSERT'

Diese Kombination ist nur bei GET in Verbindung mit USING zulässig.

63 DATEI NAME FALSCH

Der Name einer Indexdatei besteht aus maximal 8 Zeichen. Das erste Zeichen muß ein Buchstabe (A-Z) sein. Der Rest des Namens kann aus Buchstaben oder Ziffern bestehen.

64 FORMATFEHLER ZEILE/SPALTE

Formatfehler in der <LOC> Angabe.

70 DATEI NAME MEHRFACH DEFINIERT

71 VARIABLE MEHRFACH DEFINIERT

72 VARIABLE NICHT DEFINIERT

73 LETZTES UPRO NICHT BEENDET. 'EXIT' FEHLT

Das Programm wurde nicht ordnungsgemäß abgeschlossen (EXIT fehlt).

74 UPRO NICHT BEENDET. 'EXIT' FEHLT

Es wurde ein neues Unterprogramm eröffnet (ENTER), bevor das letzte Unterprogramm beendet wurde (EXIT).

75 'PERFORM' IM UPRO UNZULAESSIG

Ein PERFORM-Befehl innerhalb eines Unterprogramms ist nicht zulässig.

76 LINEARES EINTRETEN IN UPRO UNZULAESSIG

Ein Unterprogramm darf nur durch einen PERFORM-Befehl angesprochen werden. Ein lineares Eintreten in ein Unterprogramm ist nicht zulässig.

77 MAXIMAL 99 VARIABLE ZULAESSIG

78 LABEL MEHRFACH DEFINIERT

Ein Label darf nur einmal im Programm vergeben werden.

79 MEHRFACH DEFINIERTER LABEL ANGESPROCHEN

Siehe Fehler 78.

80 NICHT DEFINIERTER LABEL ANGESPROCHEN

81 'ENTER' IM UPRO FEHLT

Der Eingang eines Unterprogramms muß das folgende Format haben:

!Label ENTER.....

82 'GOTO' IN EIN UPRO NICHT MOEGLICH

83 'GOTO' ALS SPRUNG AUS UPRO NICHT ERLAUBT

84 PROGRAMM DARF MAXIMAL 255 LABELS HABEN

In einem Programm dürfen maximal 255 Label vergeben werden.

Anhang

9.2 Codetabellen

9.2.1 Ausgabecode

Soll bei der Datenausgabe auf Magnetband nicht der System-Ausgabecode (EBCDIC) verwendet werden, kann die Aufzeichnung in einem Fremdcode erfolgen.

Der Fremdcode wird unter BIBLIOTHEKEN (Supervisor-Funktionsübersicht) mit einem Code-Namen eingegeben. Bei der Datenausgabe wird der gewünschte Fremdcode unter diesem Namen aufgerufen.

Es können beliebig viele verschiedene Codes im System gespeichert werden.

Soll ein Fremdcode bei allen Ausgaben auf Magnetband benutzt werden, kann der interne Ausgabecode durch den Fremdcode überschrieben werden (siehe Supervisor-Funktion SYSTEMOPERATIONEN, MB-AUSGABECODE).

Wichtig:

Bitte beachten Sie bei der Benutzung von 7-Spur-Bändern mit gerader Parität, daß der oktale Code ØØ ein nicht schreibbares Zeichen auf MB ergibt und somit beim Wiedereinlesen in die 620 zu Fehlern führen kann. Damit ein Zeichen auf Magnetband gelesen werden kann, muß mindestens 1 Bit gesetzt sein.

Die gebräuchlichsten Codes sind:

USASCII	=	hexadezimale Verschlüsselung im USASCII
EBCDIC	=	hexadezimale Verschlüsselung im EBCDIC
NIXDORF	=	Druckzeichen auf NIXDORF-Schnelldrucker
LK-CODE	=	Lochkartencode entsprechend IBM-Locher 029
H 2000	=	Honeywell 4 x 3 "Schlangen-Code" 2000
H 6000	=	Honeywell 4 x 3 "Schlangen-Code" 6000

Anhang

USASCII	EBCDIC	DIN	IBM 1000	NIXDORF	LK-CODE					USASCII	EBCDIC	DIN	IBM 1000	NIXDORF	LK-CODE				
00	00			12		0	9	8	1	20	40								
01	01			12			9		1	21	4F	!	!	!		11		8	2
02	02			12			9		2	22	7F	"	"	"				8	7
03	03			12			9		3	23	7B	#	#	#				8	3
04	37						9		7	24	5B	$	$	$		11		8	3
05	2D					0	9	8	5	25	6C	%	%	%			0	8	4
06	2E					0	9	8	6	26	50	&	&	&	12				
07	2F					0	9	8	7	27	7D	'	'	'				8	5
08	16				11		9		6	28	4D	(	(	(	12			8	5
09	05			12			9		5	29	5D	)	)	)		11		8	5
0A	25					0	9		5	2A	5C	*	*	*		11		8	4
0B	0B			12			9	8	3	2B	4E	+	+	+	12			8	6
0C	0C			12			9	8	4	2C	6B	,	,	,			0	8	3
0D	0D			12			9	8	5	2D	60	-	-	-		11			
0E	0E			12			9	8	6	2E	4B	.	.	.	12			8	3
0F	0F			12			9	8	7	2F	61	/	/	/			0		1
10	10			12	11		9	8	1	30	F0	0	0	0			0		
11	11				11		9		1	31	F1	1	1	1					1
12	12				11		9		2	32	F2	2	2	2					2
13	13				11		9		3	33	F3	3	3	3					3
14	3C						9	8	4	34	F4	4	4	4					4
15	3D						9	8	5	35	F5	5	5	5					5
16	32						9		2	36	F6	6	6	6					6
17	26					0	9		6	37	F7	7	7	7					7
18	18				11		9	8		38	F8	8	8	8				8	
19	19				11		9	8	1	39	F9	9	9	9			9		
1A	3F						9	8	7	3A	7A	:	:	:				8	2
1B	27					0	9		7	3B	5E	;	;	;		11		8	6
1C	1C				11		9	8	4	3C	4C	<		<	12			8	4
1D	1D				11		9	8	5	3D	7E	=	=	=				8	6
1E	1E				11		9	8	6	3E	6E	>		>			0	8	6
1F	1F				11		9	8	7	3F	6F	?	?	?			0	8	7

Anhang

USASCII	EBCDIC	DIN	IBM 1000	NIXDORF	LK-CODE								USASCII	EBCDIC	DIN	IBM 1000	NIXDORF	LK-CODE							
40	7C	@	@	@					8	4			60	79	´								8	1	
41	C1	A	A	A	12					1			61	81	a			12		0			1		
42	C2	B	B	B	12					2			62	82	b			12		0			2		
43	C3	C	C	C	12					3			63	83	c			12		0			3		
44	C4	D	D	D	12					4			64	84	d			12		0			4		
45	C5	E	E	E	12					5			65	85	e			12		0			5		
46	C6	E	E	E	12					6			66	86	f			12		0			6		
47	C7	G	G	G	12					7			67	87	g	g		12		0			7		
48	C8	H	H	H	12				8				68	88	h			12		0		8			
49	C9	I	I	I	12			9					69	89	i			12		0	9				
4A	D1	J	J	J		11				1			6A	91	j			12	11				1		
4B	D2	K	K	K		11				2			6B	92	k			12	11				2		
4C	D3	L	L	L		11				3			6C	93	l	l		12	11				3		
4D	D4	M	M	M		11				4			6D	94	m	m		12	11				4		
4E	D5	N	N	N		11				5			6E	95	n			12	11				5		
4F	D6	O	O	O		11				6			6F	96	o			12	11				6		
50	D7	P	P	P		11				7			70	97	p			12	11				7		
51	D8	Q	Q	Q		11		8					71	98	q			12	11			8			
52	D9	R	R	R		11	9						72	99	r			12	11		9				
53	E2	S	S	S			0			2			73	A2	s				11	0			2		
54	E3	T	T	T			0			3			74	A3	t	t			11	0			3		
55	E4	U	U	U			0			4			75	A4	u				11	0			4		
56	E5	V	V	V			0			5			76	A5	v				11	0			5		
57	E6	W	W	W			0			6			77	A6	w				11	0			6		
58	E7	X	X	X			0			7			78	A7	x				11	0			7		
59	E8	Y	Y	Y			0		8				79	A8	y				11	0		8			
5A	E9	Z	Z	Z			0	9					7A	A9	z				11	0	9				
5B	4A	{		{	12				8	2			7B	C0	{			12		0					
5C	E0	\		\			0		8	2			7C	6A	|			12				8	7		
5D	5A	}		}		11	0			1			7D	D0	}				11	0					
5E	5F	^		^		11			8	7			7E	A1	¯			12	11						
5F	6D	_	_	_			0		8	5			7F	07				12			9		7		

Anhang

USASCII	EBCIDIC	DIN	IBM 1000	NIXDORF	LK-CODE					USASCII	EBCIDIC	DIN	IBM 1000	NIXDORF	LK-CODE				
80	20				11	0	9	8	1	A0	41				12		0	9	
81	21					0	9		1	A1	42				12		0	9	2
82	22					0	9		2	A2	43				12		0	9	3
83	23					0	9		3	A3	44				12		0	9	4
84	24					0	9		4	A4	45				12		0	9	5
85	15				11		9		5	A5	46				12		0	9	6
86	06	§		12			9		6	A6	47				12		0	9	7
87	17				11		9		7	A7	48				12		0	9	8
88	28					0	9	8		A8	49				12				8 1
89	29					0	9	8	1	A9	51				12	11		9	1
8A	2A					0	9	8	2	AA	52				12	11		9	2
8B	2B					0	9	8	3	AB	53				12	11		9	3
8C	2C					0	9	8	4	AC	54				12	11		9	4
8D	09			12			9	8	1	AD	55				12	11		9	5
8E	08			12			9	8	2	AE	56				12	11		9	6
8F	18				11		9	8	3	AF	57				12	11		9	7
90	30			12	11	0	9	8	1	B0	58				12	11		9	8
91	31						9		1	B1	59					11			8 1
92	1A				11		9	8	2	B2	62					11	0	9	2
93	33						9		3	B3	63					11	0	9	3
94	34						9		4	B4	64					11	0	9	4
95	35						9		5	B5	65					11	0	9	5
96	36						9		6	B6	66					11	0	9	6
97	08			12			9	8		B7	67					11	0	9	7
98	38						9	8		B8	68					11	0	9	8
99	39						9	8	1	B9	69						0		8 1
9A	3A						9	8	2	BA	70				12	11	0		
9B	3B						9	8	3	BB	71				12	11	0	9	1
9C	04			12			9		4	BC	72				12	11	0	9	2
9D	14				11		9		4	BD	73				12	11	0	9	3
9E	3E						9	8	6	BE	74				12	11	0	9	4
9F	E1				11	0	9		1	BF	75				12	11	0	9	5

	Anhang

USASCII	EBCIDIC DIN IBM 1000 NIXDORF	LK-CODE								USASCII	EBCIDIC DIN IBM 1000 NIXDORF	LK-CODE							
C0	76			12	11	0	9		6	E0	48			12	11	0		8	
C1	77			12	11	0	9		7	E1	B9			12	11	0	9		
C2	78			12	11	0	9	8		E2	BA			12	11	0		8	2
C3	80			12		0		8	1	E3	BB			12	11	0		8	3
C4	8A			12		0		8	2	E4	BC			12	11	0		8	4
C5	8B			12		0		8	3	E5	BD			12	11	0		8	5
C6	8C			12		0		8	4	E6	BE			12	11	0		8	6
C7	8D			12		0		8	5	E7	BF			12	11	0		8	7
C8	8E			12		0		8	6	E8	CA			12		0	9	8	2
C9	8F			12		0		8	7	E9	CB			12		0	9	8	3
CA	90			12	11			8	1	EA	CC			12		0	9	8	4
CB	9A			12	11			8	2	EB	CD			12		0	9	8	5
CC	9B			12	11			8	3	EC	CE			12		0	9	8	6
CD	9C			12	11			8	4	ED	CF			12		0	9	8	7
CE	9D			12	11			8	5	EE	DA			12	11		9	8	2
CF	9E			12	11			8	6	EF	DB			12	11		9	8	3
D0	9F			12	11			8	7	F0	DC			12	11		9	8	4
D1	A0				11	0		8	1	F1	DD			12	11		9	8	5
D2	AA				11	0		8	2	F2	DE			12	11		9	8	6
D3	AB				11	0		8	3	F3	DF			12	11		9	8	7
D4	AC				11	0		8	4	F4	EA				11	0	9	8	2
D5	AD				11	0		8	5	F5	EB				11	0	9	8	3
D6	AE				11	0		8	6	F6	EC				11	0	9	8	4
D7	AF				11	0		8	7	F7	ED				11	0	9	8	5
D8	B0			12	11	0		8	1	F8	EE				11	0	9	8	6
D9	B1			12	11	0			1	F9	EF				11	0	9	8	7
DA	B2			12	11	0			2	FA	FA			12	11	0	9	8	2
DB	B3			12	11	0			3	FB	FB			12	11	0	9	8	3
DC	B4			12	11	0			4	FC	FC			12	11	0	9	8	4
DD	B5			12	11	0			5	FD	FD			12	11	0	9	8	5
DE	B6			12	11	0			6	FE	FE			12	11	0	9	8	6
DF	B7			12	11	0			7	FF	FF			12	11	0	9	8	7

9.2.2 Interne Lochkartentabelle

	EBCDI-Code	Zeichen	Lochung
1	4-0	Space	keine
2	F-0	0	0
3	F-1	1	1
4	F-2	2	2
5	F-3	3	3
6	F-4	4	4
7	F-5	5	5
8	F-6	6	6
9	F-7	7	7
10	F-8	8	8
11	F-9	9	9
12	C-1	A	12-1
13	C-2	B	12-2
14	C-3	C	12-3
15	C-4	D	12-4
16	C-5	E	12-5
17	C-6	F	12-6
18	C-7	G	12-7
19	C-8	H	12-8
20	C-9	I	12-9
21	D-1	J	11-1
22	D-2	K	11-2
23	D-3	L	11-3
24	D-4	M	11-4
25	D-5	N	11-5
26	D-6	O	11-6
27	D-7	P	11-7
28	D-8	Q	11-8
29	D-9	R	11-9
30	E-2	S	0-2
31	E-3	T	0-3
32	E-4	U	0-4
33	E-5	V	0-5
34	E-6	W	0-6
35	E-7	X	0-7
36	E-8	Y	0-8
37	E-9	Z	0-9
38	4-A	¢	12-2-8
39	4-B	.	12-3-8
40	4-C		12-4-8
41	4-D	(	12-5-8
42	4-E	+	12-6-8
43	4-F	!	12-7-8
44	5-0	&	12

Anhang

	EBCDI-Code	Zeichen	Lochung
45	5-A	!	11-2-8
46	5-B	$ (Ö)	11-3-8
47	5-C	*	11-4-8
48	5-D	)	11-5-8
49	5-E	;	11-6-8
50	5-F	¬	11-7-8
51	6-0	-	11
52	6-1	/	0-1
53	6-B	,	0-3-8
54	6-C	%	0-4-8
55	6-D	_	0-5-8
56	6-E	>	0-6-8
57	6-F	?	0-7-8
58	7-A	:	2-8
59	7-B	# (Ä)	3-8
60	7-C	@ (Ü)	4-8
61	7-D	'	5-8
62	7-E	=	6-8
63	7-F	"	7-8
64	8-0		12-0-1-8
65	8-1		12-0-1
66	8-2		12-0-2
67	8-3		12-0-3
68	8-4		12-0-4
69	8-5		12-0-5
70	8-6		12-0-6
71	8-7		12-0-7
72	8-8		12-0-8
73	8-9		12-0-9
74	8-A		12-0-2-8
75	8-B		12-0-3-8
76	8-C		12-0-4-8
77	8-D		12-0-5-8
78	8-E		12-0-6-8
79	8-F		12-0-7-8
80	9-0		12-11-1-8
81	9-1		12-11-1
82	9-2		12-11-2
83	9-3		12-11-3
84	9-4		12-11-4
85	9-5		12-11-5
86	9-6		12-11-6
87	9-7		12-11-7
88	9-8		12-11-8

Anhang

	EBCDI-Code	Zeichen	Lochung
89	9-9		12-11-9
90	9-A		12-11-8-2
91	9-B		12-11-8-3
92	9-C		12-11-8-4
93	9-D		12-11-8-5
94	9-E		12-11-8-6
95	9-F		12-11-8-7
96	C-0	+ 0	12-0
97	D-0	- 0	12-0

Anhang

9.2.3 Honeywell-4x3-Code (Schlangen-Code)

Diese Methode bietet die Möglichkeit, einen Oktal-Ausgabecode (6-Bit-Zeichen plus 1 Paritätsbit) in gepackter Form auf 9-Spur-Magnetbändern aufzuzeichnen.

Auf einem 7-Spur-Magnetband belegt ein 6-Bit-Zeichen plus Paritätsbit 1 Sprosse. Vier 6-Bit-Zeichen belegen also 4 Sprossen (Abb. 1).

Auf einem 9-Spur Magnetband nutzt der 4x3-Code alle 8 Datenbits aus. Daraus ergibt sich eine gepackte Aufzeichnung von vier 6-Bit-Zeichen auf 3 Sprossen. Daher die Bezeichnung "4x3-Code" (Abb. 2), (Abb. 3).

Abb. 1: 7-Kanal-Magnetband (4-Zeichen-Darstellung)

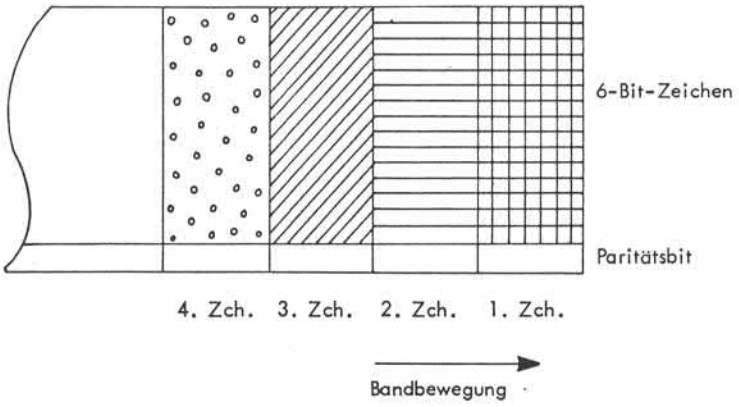


Abb. 2: 9-Kanal-Magnetband (4-Zeichen-Darstellung im 4x3-Code) H 6000

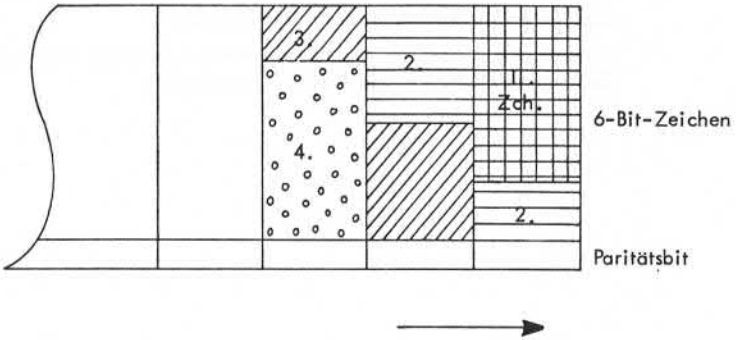
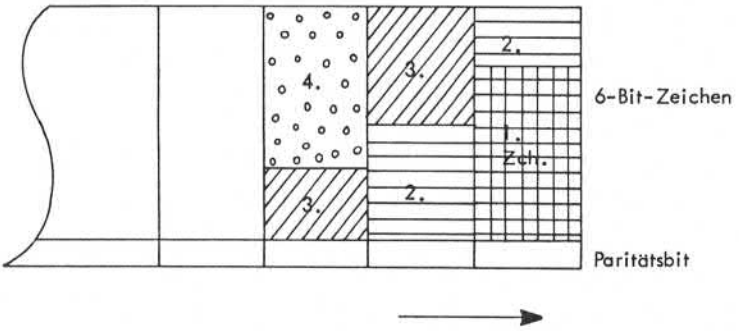


Abb. 3: 9-Kanal-Magnetband (4-Zeichen-Darstellung im 4x3-Code) H 2000



	Anhang

Um zu gewährleisten, daß beim Einlesen die Anzahl der gelesenen und entpackten Zeichen mit der Anzahl der geschriebenen und gepackten Zeichen übereinstimmt, muß darauf geachtet werden, daß die Anzahl Zeichen des auszugebenden Bandsatzes durch 4 teilbar ist.

Soll mit dem 4x3-Code gearbeitet werden, müssen die entsprechenden Zeichen als AUSGABECODE gespeichert werden.

Anhang

MAGNETBAND-AUSGABECODE-TABELLE

TASTATUR-ZEICHEN	EBCDIC hexa	EBCDIC oktal	H 2000 oktal	H 6000 oktal
A	C1	301	21	21
B	C2	302	22	22
C	C3	303	23	23
D	C4	304	24	24
E	C5	305	25	25
F	C6	306	26	26
G	C7	307	27	27
H	C8	310	30	30
I	C9	311	31	31
J	D1	321	41	41
K	D2	322	42	42
L	D3	323	43	43
M	D4	324	44	44
N	D5	325	45	45
O	D6	326	46	46
P	D7	327	47	47
Q	D8	330	50	50
R	D9	331	51	51
S	E2	342	62	62
T	E3	343	63	63
U	E4	344	64	64
V	E5	345	65	65
W	E6	346	66	66
X	E7	347	67	67
Y	E8	350	70	70
Z	E9	351	71	71
0	F0	360	00	00
1	F1	361	01	01
2	F2	362	02	02
3	F3	363	03	03

Anhang

TASTATUR-ZEICHEN	EBCDIC hexa	EBCDIC oktal	H 2000 oktal	H 6000 oktal
4	F4	364	04	04
5	F5	365	05	05
6	F6	366	06	06
7	F7	367	07	07
8	F8	370	10	10
9	F9	371	11	11
SP	40	100	15	20
¢	4A	112	77	37
.	4B	113	33	33
<	4C	114	60	36
(	4D	115	74	35
+	4E	116	20	60
|	4F	117	75	12
&	50	120	17	32
!	5A	132	57	77
$	5B	133	53	53
*	5C	134	54	54
)	5D	135	34	55
;	5E	136	32	56
¬	5F	137	56	40
-	60	140	40	52
/	61	141	61	61
≠	6A	152	56	
.	6B	153	73	34
%	6C	154	35	73
_	6D	155	76	72
>	6E	156	16	16
?	6F	157	37	17
:	7A	172	14	15

Anhang

TASTATUR-ZEICHEN	EBCDIC hexa	EBCDIC oktal	H 2000 oktal	H 6000 oktal
#	7B	173	52	13
@	7C	174	72	14
'	7D	175	12	57
=	7E	176	13	75
"	7F	177	55	76

9.2.4 Druckercodetabellen

Im System gibt es nur eine Durckercodetabelle. Damit nicht für unterschiedliche Drucker unterschiedliche Betriebssysteme erstellt werden müssen, gibt es die Möglichkeit, die Druckercodetabelle als externe Codetabelle zu erstellen. Die Codetabelle wird in der Supervisorebene unter der Funktion BIBLIOTHEKEN in das System eingegeben und anschließend als Systemdrucker-Tabelle zugewiesen.

Nachfolgend die gebräuchlichen Codetabellen:

Anhang

		A	B	C	D	E	F	G	H	I	J
A = Tastatur-	blank	40	20								
zeichen	A	C1	41								
	B	C2	42								
B = Interncode	C	C3	43								
	D	C4	44								
C = Standard für	E	C5	45								
Centronics-Dr.	F	C6	46								
	G	C7	47								
D = Centronics-Dr.	H	C8	48								
f.Dänem./	I	C9	49								
Norwegen	J	D1	4A								
	K	D2	4B								
E = Centronics-Dr.	L	D3	4C								
f.Schweden/	M	D4	4D	wie C	wie C	wie C	wie C	wie C	wie C	wie C	wie C
Finnland	N	D5	4E								
	O	D6	4F								
F = Nadeldrucker	P	D7	50								
150 Z/s Zeichen-	Q	D8	51								
generator	R	D9	52								
056460.3.15	S	E2	53								
	T	E3	54								
G = Data-Products-	U	E4	55								
Drucker, Walze	V	E5	56								
PN 240160-042	W	E6	57								
	X	E7	58								
H = Data-Products-	Y	E8	59								
Drucker Walze	Z	E9	5A								
PN 239016/240119											

I = Data-Products-
Drucker, Walze
PN 239114/240121

J = Data-Products-
Drucker, Walze
PN 240160-078

Anhang

A	B	C	E	F	G	H	I	J
0	F0	30						
1	F1	31						
2	F2	32						
3	F3	33						
4	F4	34						
5	F5	35						
6	F6	36						
7	F7	37						
8	F8	38						
9	F9	39						
¢	4A	5B		5E				20
.	4B	2E						
<	4C	3C						
C	4D	28						
+	4E	2B						
▮	4F	5D		DB				60
&	50	26						
!	5A	21						
$	5B	24						
*	5C	2A						
)	5D	29						
;	5E	3B						
¬	5F	20		DA	5C	5C	5C	20
-	60	2D						
/	61	2F						
≠	6A	5E			DC			7F
,	6B	26						
%	6C	25						
_	6D	5F						
>	6E	3E						
?	6F	3F						
:	7A	3A						
#	7B	23						
@	7C	40						
'	7D	27						
=	7E	3D						
"	7F	22						

A	B	C	D	E	F	G	H	I	J
a	81	41			A1				
b	82	42			A2				
c	83	43			A3				
d	84	44			A4				
e	85	45			A5				
f	86	46			A6				
g	87	47			A7				
h	88	48			A8				
i	89	49			A9				
j	91	4A			AA				
k	92	4B			4B				
l	93	4C			4C				
m	94	4D			AD				
n	95	4E			AE				
o	96	4F			AF				
p	97	50			B0				
q	98	51			B1				
r	99	52			B2				
s	A2	53			B3				
t	A3	54			B4				
u	A4	55			B5				
v	A5	56			B6				
w	A6	57			B7				
x	A7	58			B8				
y	A8	59			B9				
z	A9	5A			BA				
Ä	AD	20	C8	C8	5B				
Ö	A0	20	C5	C5	5C				
Ü	BD	20	C7	C7	5D				
ä	9F	20			BB				
ö	8B	20			BC				
ü	9B	20			BD				
ß	A1	20			BE				
? +0	C0	3F			3F				
! -0	D0	21			21				
# FKZ	FB	23			23				

Anhang

A	B	C	D	E	F	G	H	I	J
§					5E				
e'					BF				
OCR									
0					C0				
1					C1				
2					C2				
3					C3				
4					C4				
5					C5				
6					C6				
7					C7				
8					C8				
9					C9				
Г					CA				
Ч					CB				
Л					CC				
Ш					CD				
ø					CE				
ъ					CF				

A = Zeichen auf Drucker

	Anhang

9.3 Berechnung der Plattenbelegung

9.3.1 Stapel

Zusätzlich zu den in einem Stapel enthaltenen Anwender-Daten schreibt das System folgende Merker und Zeigerfelder auf die Magnetplatte:

- Pro Stapel:
  - 5 Byte Zeigerfeld nach dem letzten Satz
  - 1 Byte Stapelanfang
  - 1 Byte Stapelende
  - 256 Byte File Information Block
- Pro Satz:
  - 1 Byte Satzmarke
  - 5 Byte Zeigerfeld
- Pro Feld
  - 1 Byte Feldbegrenzung

Der tatsächliche Platzbedarf eines Stapels wird daher wie folgt berechnet:

$$S_B = R\ (F_B + f + 6) + 263$$

S_B = Platzbedarf in Byte

R = Anzahl Sätze

F_B = Satzlänge in Byte

f = Anzahl Felder im Satz

Beispiel:

Ein Stapel mit 10.000 Sätzen wird in das System eingegeben. Die Satzlänge ist 45 Byte. Jeder Satz enthält 7 Felder.

$R = 10.000$

$F_B = 45$

$f = 7$

$SB = 10.000\ (45 + 7 + 6) + 263$
$= 10.000\ (58) + 263$
$= 580.000 + 263$
$= 580.263$ Byte

Jeder Sektor der Magnetplatte ist 256 Bytes lang, jedoch stehen nur 242 Bytes zur Datenspeicherung zur Verfügung. Die Plattenbelegung in Sektoren wird daher nach folgender Formel berechnet:

$$S_S = \frac{S_B}{242}$$

Das Ergebnis wird auf die nächste höhere Ganzzahl gerundet.

Beispiel:

$$S_S = \frac{580.263}{242} = 2397.8 = 2398 \text{ Sektoren}$$

Das Beispiel geht davon aus, daß nur ein Eingabeformat für den Stapel verwendet wurde. Werden mehrere Formate verwendet, muß die Plattenbelegung für jedes Format einzeln berechnet und die Ergebnisse addiert werden.

Die Anzahl Segmente, die auf einen 33 MB-Platteneinheit zur Verfügung stehen, beträgt 96.640 auf der ersten und 129.280 auf der zweiten Einheit.

9.3.2 Indexdatei

Jede Indexdatei besteht aus dem Index und einem zugehörigen Datenstapel. Der Platzbedarf des Datenstapels wird nach dem in Abschnitt 9.3.1 beschriebenen Verfahren berechnet.

Der Platzbedarf des Index wird einerseits durch Typ und Länge des Schlüssels und andererseits durch verschiedene Zeigerfelder, die vom System zu jedem Schlüssel hinzugefügt werden, bestimmt.

Der erste Schritt ist die Errechnung der Schlüssellänge.

	Anhang

Abhängig vom Schlüsseltyp gelten folgende Formeln:

- Schlüsseltyp "E": $K = \ell$
- Schlüsseltyp "A": $K = \frac{2 \cdot \ell}{3}$
- Schlüsseltyp "N": $K = \frac{\ell}{2}$

ℓ = Anzahl Zeichen im Schlüsselbegriff

Ist das Ergebnis keine durch 2 teilbare Ganzzahl, muß es auf die nächst höhere, durch 2 teilbare Ganzzahl aufgerundet werden.

Beispiele:

- Schlüsseltyp "N", $\ell = 6$

  $\frac{6}{2} = 3$, aufgerundet K = 4

- Schlüsseltyp "E", $\ell = 5$

  aufgerundet K = 6

- Schlüsseltyp "A", $\ell = 7$

  $\frac{2 \cdot 7}{3} = 4\frac{2}{3}$, aufgerundet K = 6

Zur errechneten Schlüssellänge müssen folgende Zeigerfelder addiert werden:

- 4 Byte Adresse des verbundenen Datensatzes
- 2 Byte Adresse des File Information Block
- 1 Byte Zeiger auf den folgenden Indexeintrag
- 1 Byte Zeiger auf den vorhergehenden Indexeintrag

Die Länge eines Eintrags ist also:

$P_\ell = K + 8$

Anhang

Im nächsten Schritt wird die Anzahl Einträge pro Sektor berechnet:

$$P_S = \frac{242}{P_\ell}$$

Da ein Eintrag die Sektorengrenze nicht überschreiten kann, wird das Ergebnis auf die nächst niedrigere Ganzzahl abgerundet.

Beispiel:

Ein Stapel enthält 10.000 Sätze. Im Index soll für jeden Satz ein Eintrag angelegt werden. Die Schlüssellänge beträgt 8 Stellen, der Schlüsseltyp ist "A".

- $K = \frac{2 \cdot \ell}{3} = \frac{2 \cdot 8}{3} = 5\frac{1}{3} \rightarrow 6$
- $P_e = K + 8 = 6 + 8 \rightarrow 14$
- $P_s = \frac{242}{P_\ell} = \frac{242}{14} = 17{,}3 \rightarrow 17$

In jedem Sektor sind 17 Einträge enthalten.

Der Platzbedarf einer Indexstufe in Sektoren wird errechnet, indem die Anzahl Einträge im Index durch die Anzahl Einträge pro Sektor dividiert wird.

$$L = \frac{N}{P_s}$$

Das Ergebnis wird auf die nächst höhere Ganzzahl aufgerundet.

Beispiel:

$$L = \frac{N}{P_s} = \frac{10.000}{17} = 588{,}2 = 589$$

Die Größe der nächsten Indexstufe wird errechnet, indem die Anzahl Sektoren der vorhergehenden Indexstufe durch die Anzahl Einträge pro Sektor dividiert wird. Der Rechengang wird solange fortgesetzt, bis eine Indexstufe erreicht ist, die nur noch einen oder zwei Sektoren belegt.

Beispiel:

$$\frac{589}{17} = 34{,}7 \rightarrow 35$$

$$\frac{35}{17} = 2{,}1 \rightarrow 3$$

$$\frac{3}{17} = 0{,}1 \rightarrow 1$$

Der Platzbedarf des gesamten Index ergibt sich aus der Addition der belegten Sektoren aller Indexstufen.

Beispiel:

589 + 35 + 3 + 1 = 628

9.3.3 Sortierter Stapel

Bedingt durch das verwendete Sortierverfahren ist der Platzbedarf eines sortierten Stapels während des Sortierens größer als nach abgeschlossener Sortierung.

Der erste Schritt ist die Errechnung der Schlüssellänge K. Das Verfahren wird in Abschnitt 9.3.2 beschrieben.

Zur ermittelten Schlüssellänge ist ein Zeigerfeld von 6 Byte zu addieren. Die Länge eines Eintrags ist also:

$$P_\ell = K + 6$$

Die Anzahl Einträge pro Sektor wird mit folgender Formel ermittelt:

$$P_s = \frac{242}{P_\ell}$$

Das Ergebnis muß auf die nächst niedrigere Ganzzahl abgerundet werden.

Die Anzahl belegter Sektoren wird ermittelt, indem die Anzahl zu sortierender Sätze durch die Anzahl der Einträge pro Sektor dividiert wird.

$$S = \frac{N}{P_s}$$

Das Ergebnis muß auf die nächst höhere Ganzzahl aufgerundet werden.

Beispiel:

Ein Stapel mit 10.000 Sätzen soll sortiert werden. Der Schlüssel ist 10 Stellen lang, das Sortierverfahren ist EBCDIC.

$$K = 10$$

$$P_\ell = 10 + 6 = 16$$

$$P_s = \frac{242}{16} = 15{,}13 \rightarrow 15$$

$$S = \frac{10.000}{15} = 666{,}67 = 667$$

Anhang

Da während des Sortierens drei Arbeitsdateien eröffnet werden, ist die ermittelte Anzahl belegter Sektoren mit 3 zu multiplizieren.

Beispiel:

$$Tmax. = 3 \cdot S = 3.667 = 2001$$

Der Platzbedarf nach abgeschlossener Sortierung wird mit folgender Formel berechnet:

$$F_s = \frac{N}{60}$$

Das Ergebnis wird auf die nächst höhere Ganzzahl aufgerundet.

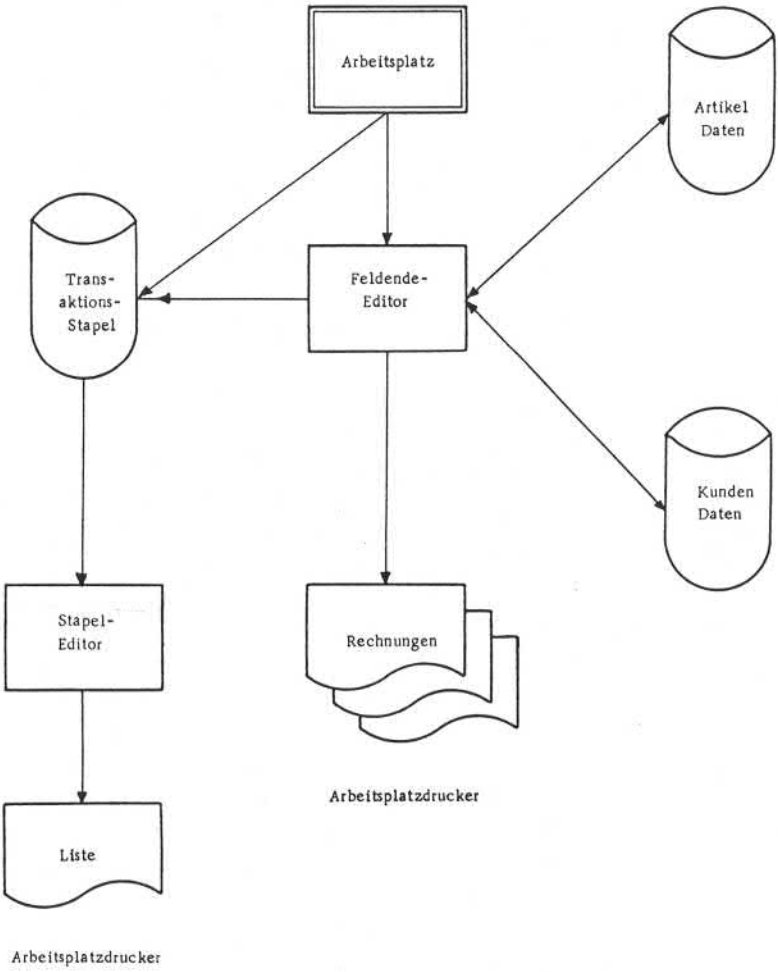
Beispiel:

$$F_s = \frac{N}{60} = \frac{10.000}{60} = 166{,}67 \rightarrow 167$$

9.4 Programmbeispiel

Das folgende Programmbeispiel zeigt eine Echtzeit-Fakturierung mit Ausdruck einer Rechnungsausgangsliste durch ein nachfolgendes Stapel-Editorprogramm. Das Beispiel erhebt keinerlei Anspruch auf Vollständigkeit, es sollen lediglich die Möglichkeiten der Dateiverwaltung des Systems dargestellt werden.

	Anhang


Arbeitsplatz
Artikel Daten
Trans-aktions-Stapel
Felde nde-Editor
Kunden Daten
Stapel-Editor
Rechnungen
Arbeitsplatzdrucker
Liste
Arbeitsplatzdrucker

	Anhang

Ablauf:

- Zum Tagesanfang wird das Fakturierprogramm durch Anwahl der Funktion DATEI-BEARBEITUNG und des entsprechenden Standard-Jobs gestartet.
- Erscheint ein Kunde, wird nach Eingabe der Kundennummer der zugehörige Kundenstammsatz gelesen und der Rechnungskopf gedruckt.
- Die Bedienungskraft gibt Nummer und Menge der gewünschten Artikel ein. Das System prüft anhand der Artikelstammdatei die Gültigkeit der Artikelnummer und den Lagerbestand. Ist der Artikel vorhanden, wird sofort die Rechnungszeile gedruckt und die Menge vom Lagerbestand abgebucht.
- Nach Fakturierung aller Artikelpositionen druckt das System automatisch den Rechnungsabschluß. Ein ggf. einzuräumender Rabatt ist im Kundenstammsatz gespeichert. Der Jahresumsatz des Kunden wird im Kundenstammsatz aktualisiert.
- Nach Beendigung des Fakturierprogramms wird mit Hilfe eines Ausgabeprogramms aus den im Transaktionsstapel gespeicherten Daten die Rechnungsausgangsliste gedruckt.

Anhang

System 620/45 — **Eingabe-Format** — NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name K U S T A M M Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit # beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Füllen	Kurz-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füllzeich: Mit Daten	Füllzeich: Ohne Daten	Step. Som.: Add./Sub.	Step. Som.: Nummer	Tabelle: Nummer	Tabelle: Lng. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-funktion	Prof-Ziffer: Nummer	Prof-Ziffer: Methode	Prof-Ziffer: Mehr. Felder	Update	Auftrag. Folge	Tab.-Feld	Zusatz-Feld	Kennung	Feldenda-Editor	Feld enf. /löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	24	25	26	27	28	29	30	31	32	33
0	1	1	KUNDEN-STAMMDATEN																											
1	3	1	KD-NR.: #	6	N			Y	Y																					Y
2	3	17	LOESCHEN? J/N: #	1	L																									Y
3	4	1	NAME 1: #	30	L																									
4	5	1	NAME 2: #	30	L																									
5	6	1	STR.: #	30	L																									
6	8	1	ORT: #	30	L																									
7	9	1	RABATT: #	3	N					Y	Y																			Y
8	14	1	UMSATZ IN DM: #	8	N					Y	Y																			

Feld-Nr.		Min-Wert			Max-Wert

Anhang

K U N D E N - S T A M M D A T E N

Kd.Nr.	Anschrift	Rabatt	Umsatz
202000	Herr Xaver Abelein Talstr. 15 3164 Hohenhameln	9,0%	5.789.638
202600	Firma Eisen & Stahl Halbergstr. 41 6600 Saarbruecken	3,0%	8.935.940
203000	Frl. Walburga Bendix Auf dem Meere 55 7000 Stuttgart	5,0%	478.592
204000	Firma Reinhardt Woop Kreuzbergerstr. 15 6349 Fleisbach	12,0%	1.379.600
204100	Emil-Versand Ziegenwald 3579 Schwarzenborn	1,5%	91.583

NIXDORF COMPUTER

System 620/45

Eingabe-Format

Organisator _____ Eingabe-Format-Name ARTSTAMM Seite _____ von _____ Datum _____

Feld-Nr.	Zeile	Masken-Beginn / Spalte	Maske mit # beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Folge	Kor.-Folge	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füll-zeich	Stapel-Sum.	Tabelle	Ausg.-Feld	Ausg.-Folge	Grenzprüfg.	Aufo-Funktion	Prüf-Ziffer	Nummer	Methode	Mehr. Felder	Update	Auftrag. Folge	Tab.-Feld	Zusatz-Feld	Kennung	Feldende-Editor	Feld anf. /löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	1	1	ARTIKEL-STAMMDATEN																											
1	3	1	ART.-NR.: #	6	N			Y	Y																			Y	Y	
2	3	19	LOESCHEN?: #	1	L																									
3	5	1	ART.-BEZ.: #	28	L																									
4	7	1	PREIS: #	6	N						Y																	Y		
5	7	19	BESTAND: #	8	N						Y																			

Feld-Nr.		Zl.	Min-Wert		22	Max-Wert

Anhang

A R T I K E L - S T A M M D A T E N

Art.Nr.	Art.Bezeichnung	Preis	Bestand
410670	Notstrom-Aggregat 1500 W	1.290,00	947
436930	Schweissgeraet 220 V/125 A	750,00	923
578441	Schlagbohrmaschine 4-Gang	259,00	1.018
588220	Saegeblatt-Sortiment	6,75	0
588230	Saegeblatt	9,50	0
588428	Bohrstaender	45,00	930
588473	Elektronikschrauber	279,00	913
588638	Maschinenschraubstock	19,90	11.970
589013	Sortiment Schleifbaender	14,90	985
589272	Wanknuteinrichtung	49,75	987
589329	Schraubendreher-Klingensatz	19,90	990
640878	Schlagbohrmaschine 2-Gang	198,00	975
640981	Radialarmsaege	875,00	992
642289	Einhand-Hobel EH 822	369,00	989
642540	Oberfraese DN 65	115,00	419
644058	Kompressor-Spritzanlage	259,00	998
648081	Werkbank "Workmate"	198,00	998
648514	Handkreissaege-Vorsatz	47,50	985
651234	Schweisselektroden	12,30	917
652300	Bohrhammer	125,00	993
661234	Schleifblaetter sortiert	5,60	98
725610	Hartmetallbohrer 10 mm	3,60	987
795519	Stichsaege-Vorsatz	37,50	998
795541	Bandschleifer-Vorsatz	69,00	997
795610	Schleifbock-Vorsatz	45,00	977

System 620

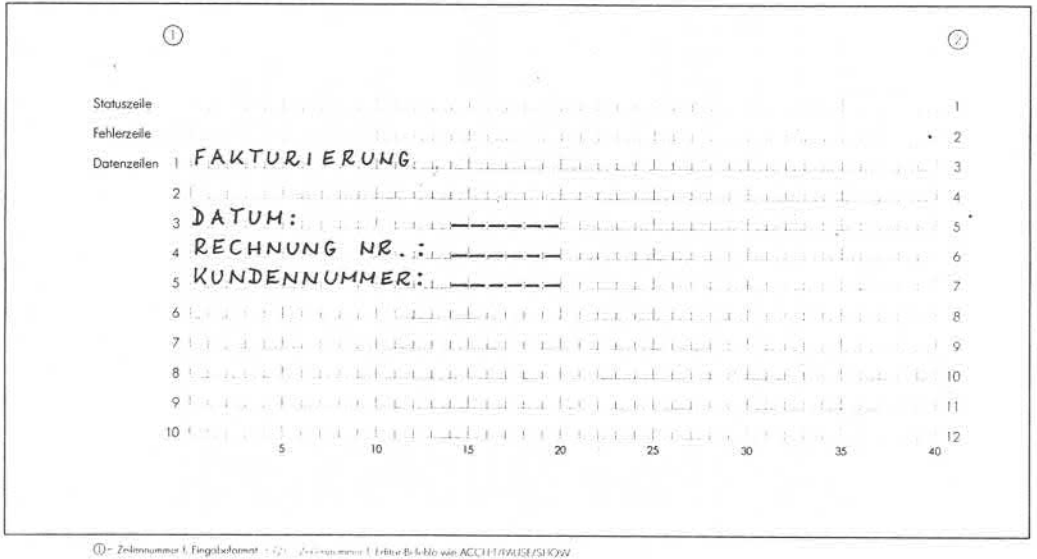
Display-Aufbau 480/360 Zeichen

NIXDORF COMPUTER

Firma _____

Organisator _____ Name des Eingabe-Formats FAKT1 Datum _____

Seite: _____ von: _____



Z 9x5.00, 8/77

① = Zeilennummer f. Eingabeformat · ② = Zeilennummer f. Erster Bild-No wie ACCEPT/WAUSE/SKIP/W

System 620

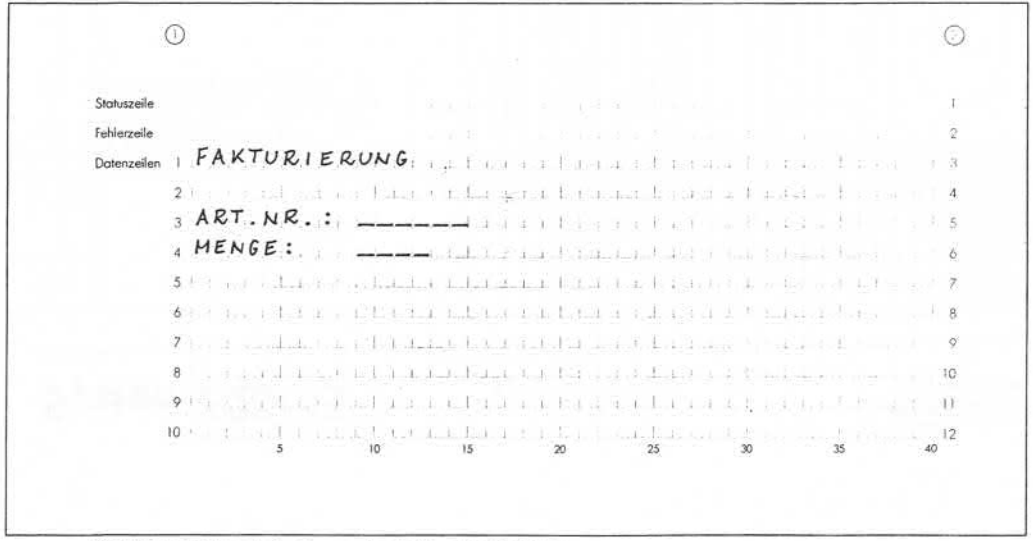
Display-Aufbau 480/360 Zeichen

NIXDORF COMPUTER

Firma _____

Organisator _____ Name des Eingabe-Formats FAKT2 Datum _____

Seite: _____ von: _____



Z 96.030 8/77

① = Zeilennummer f. Eingabeformat · ② = Zeilennummer f. Editor-Befehle wie ACCEPT/PAUSE/SHOW

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name F A K T 1 _ _ _ Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit ≠ beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Kont.-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füllzeich: Mit Daten	Füllzeich: Ohne Daten	Stap. Sum.: Add./Sub.	Stap. Sum.: Nummer	Tabelle: Nummer	Tabelle: Ung. Wert	Tabelle: Organis.	Grenzpolig.	Auto-Funktion	Prüf-Ziffer: Nummer	Prüf-Ziffer: Methode	Prüf-Ziffer: Mehr. Felder	Update	Aufzeig. Folge	Tab.-Feld	Zusatz-Feld	Kettung	Feldende-Editor	Feld einf./löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	24	25	26	27	28	29	30	31	32	33
0	1	1	FAKTURIERUNG																											
1	3	1	DATUM: ≠	6	N			Y	Y																					
2	4	1	RECHNUNG NR.: ≠	6	N						Y																			
3	5	1	KUNDENNUMMER: ≠	6	N			Y	Y																				Y	

Feld-Nr.		21	Min-Wert		22	Max-Wert

System 620/45

Eingabe-Format

NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name FAKT2 Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit # beenden (Maskenlänge max. 40 Zeichen)	Feldlänge	Feldtyp	Prüfen	Korr.-Prüfen	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füllzeich: Min Daten	Füllzeich: Ohne Daten	Stap. Sum.: Add./Sub.	Tabelle: Nummer	Tabelle: Nummer	Tabelle: Urg. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-Funktion	Prüf-Ziffer: Nummer	Prüf-Ziffer: Methode	Prüf-Ziffer: Mehr. Felder	Update	Aufsteig. Folge	Tab.-Feld	Zusatz-Feld	Kettung	Feldende-Editor	Feld entf. /löschen
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	23	24	25	26	27	28	29	30	31	32	33
0	1	1	FAKTURIERUNG																											
1			(DATUM)	6	N						Y									A											
2			(RE-NR)	6	N						Y									A											
3	3	1	ART.NR.: #	6	N			Y	Y																					
4	4	1	MENGE: #	4	N						Y																		Y	Y
5			(WERT)	7	N						Y																				
6			(RABATT)	3	N						Y																				

Feld-Nr.		21	Min-Wert		22	Max-Wert

Anhang

System 620/45 — **Eingabe-Format** — NIXDORF COMPUTER

Organisator _____ Eingabe-Format-Name F A K T 3 Seite ____ von ____ Datum _____

Feld-Nr.	Masken-Beginn: Zeile	Masken-Beginn: Spalte	Maske mit # beenden (Maskenlänge max. 40 Zeichen!)	Feldlänge	Feldtyp	Halten	Kor.-Halten	Eingabezwang	Beendigungszwang	Feldbegrenzung	Rechtsbündig	Füllzeich: Mit Daten	Füllzeich: Ohne Daten	Add/Sub.	Slop. Sum.: Nummer	Tabelle: Nummer	Tabelle: Lng. Wert	Tabelle: Organis.	Grenzprüfg.	Auto-Funktion	Prüf-Ziffer: Nummer	Prüf-Ziffer: Methode	Prüf-Ziffer: Mehr. Felder	Updatie	Aufstäg. Folge	Tab.-Feld	Zusatz-Feld	Kürzung	Feldende-Zeichen	Feld auf /löschen
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1				(DATUM)	9	N						Y									J										
2				(RE-NR)	9	N						Y									J										E
3				(WARENWERT)	9	N						Y																			
4				(RABATT)	9	N						Y																			
5				(MWST)	9	N						Y																			
6				(BRUTTO)	9	N						Y																			

Feld-Nr.		21	Min-Wert		22	Max-Wert	

Anhang

Firma
Eisen & Stahl
Halbergstr. 41

6600 Saarbruecken

R E C H N U N G Nr.: 200002 Kd.Nr.: 202600 Datum: 2.08.77

Art.Nr.	Art.Bezeichnung	Menge	Preis	Gesamt
410670	Notstrom-Aggregat 1500 W	1	1.290,00	1.290,00
436930	Schweissgeraet 220 V/125 A	2	750,00	1.500,00
651234	Schweisselektroden	50	12,30	615,00
652300	Bohrhammer	2	125,00	250,00
			Zwischensumme:	3.655,00
			3,0% Rabatt:	109,65
			Netto:	3.545,35
			+ 11% MWST:	389,99
			Brutto:	3.935,34

```
DEFINE XKUND XART.
DECLARE SUMME KDNR WERT RABATT ZEILE.
NOTE ** BLATTHOEHE EINRICHTEN **
     ** DATEIEN EROEFFNEN     **.
WHEN START
     TYPE <PAGE 72>;
     OPEN XKUND|UPD;
     OPEN XART|UPD.
WHEN NOT FMT 1
     GOTO !ZEILE.
NOTE ** PRUEFEN, OB BEI MANUELLER **
     ** ANWAHL VON FORMAT 1 DIE   **
     ** VORHERGEHENDE RECHNUNG    **
     ** ABGESCHLOSSEN WURDE       **.
IF SUMME ≠ 0
     PAUSE 'RECHNUNG ABSCHLIESSEN!!!';
     LINK 2;
     RELEASE.
NOTE ** KUNDENSATZ LESEN **.
GET XKUND USING (3)
     ELSE PAUSE 'FALSCHE KUNDENNUMMER!';
     POSITION (3).
MOVE (3) TO KDNR.
NOTE ** RECHNUNGSNUMMER ERHOEHEN **
     ** RECHNUNGSKOPF SCHREIBEN  **.
MOVE (2) + 1 TO (2).
PERFORM !KOPF.
LINK 2.
RELEASE.
```

Anhang

```
NOTE ** BEGINN ARTIKELZEILE **.
!ZEILE
WHEN NOT FMT 2
     GOTO !FUSS.
WHEN FIELD 4
     GOTO !ZEILE1.
NOTE ** ARTIKELSTAMMSATZ LESEN **.
GET XART USING (3)
     ELSE PAUSE 'ARTIKELNUMMER FALSCH!';
     POSITION (3).
RELEASE.
!ZEILE1
IF @5@ = 0
     PAUSE 'KEIN BESTAND!';
     POSITION (3).
MOVE @5@ - (4) TO WERT.
IF WERT < 0
     PAUSE 'BESTAND UNTERSCHRITTEN!';
     POSITION (3).
TYPE <SKIP 11> @1@|LZ <SKIP 19> @3@
<SKIP 49> (4)|'_.__0-'
<SKIP 56> @4@|'_._0_,__' <DEFER>.
MOVE @4@ * 10 TO WERT.
MOVE (4) * WERT TO WERT.
ADD 5 TO WERT.
DIVIDE 10 INTO WERT.
ADD WERT TO SUMME.
TYPE <SKIP 2> WERT|'__._0_,__-'.
MOVE WERT TO (5).
MOVE RABATT TO (6).
```

```
MOVE @5@ - (4) TO @5@.
ADD 1 TO ZEILE.
IF ZEILE < 30 RELEASE.
TYPE <TOP>.
NOTE ** KUNDENSTAMMSATZ LESEN **.
GET XKUND USING KDNR ELSE STOP.
PERFORM !KOPF.
RELEASE.
NOTE ** RECHNUNGSABSCHLUSS **.
!FUSS
GET XKUND USING KDNR ELSE STOP.
TYPE <SKIP 63> 12'-'.
TYPE <SKIP 48> 'Zwischensumme: '
SUMME|'_.___._0_,__-' <LF>.
MOVE SUMME TO (3).
MOVE SUMME * @7@ TO WERT.
ADD 500 TO WERT.
DIVIDE 1000 INTO WERT.
TYPE <SKIP 48> @7@|'0_,_' '% Rabatt:  '
WERT|'_.___._0_,__-'.
TYPE <SKIP 63> 12'-'.
MOVE WERT TO (4).
SUBTRACT WERT FROM SUMME.
TYPE <SKIP 48> 'Netto:' <SKIP 63>
SUMME|'_.___._0_,__-' <LF>.
MOVE SUMME + @8@ TO @8@.
MOVE SUMME * 11 TO WERT.
ADD 50 TO WERT.
DIVIDE 100 INTO WERT.
TYPE <SKIP 48> '+ 11% MWST:'
<SKIP 63> WERT|'_.___._0_,__-'.
```

	Anhang

```
TYPE <SKIP 63> 12'-'.
MOVE WERT TO (5).
ADD WERT TO SUMME.
TYPE <SKIP 48> 'Brutto:'
<SKIP 63> SUMME|'_.___._0_,__-'.
TYPE <SKIP 63> 12'='.
MOVE SUMME TO (6).
TYPE <TOP>.
MOVE 0 TO SUMME.
LINK 1.
RELEASE.
NOTE ** BEGINN RECHNUNGSKOPF **.
!KOPF ENTER
TYPE <SKIP 11> @3@.
TYPE <SKIP 11> @4@.
TYPE <SKIP 11> @5@.
TYPE <LF> <SKIP 11> @6@.
TYPE <LF> <LF> <LF> <LF> <LF> <SKIP 11>
'R E C H N U N G  Nr.: ' (2)|LZ
<SKIP 42> 'Kd.Nr.: ' KDNR|'____0' <SKIP
 60> 'Datum: ' (1)|'0_.__.__'.
TYPE <SKIP 11> 64'-'.
TYPE <SKIP 11> 'Art.Nr.  Art.Bezeichnung
' <SKIP 49> 'Menge' <SKIP 59> 'Preis'
<SKIP 69> 'Gesamt'.
TYPE <SKIP 11> 64'-' <LF>.
MOVE 1 TO ZEILE.
EXIT.
```

R E C H N U N G S - A U S G A N G S B U C H

Kto.Nr.	Re.-Nr.	Datum	Warenwert	Rabatt	MwSt	Brutto
202000	100001	2.08.77	12.900,00	1.161,00	1.291,29	13.030,29
202600	100002	2.08.77	1.500,00	45,00	160,05	1.615,05
203000	100003	2.08.77	5.179,99-	258,99-	541,30-	5.462,30-
204000	100004	2.08.77	1.350,00	162,00	130,68	1.318,68
204100	100005	2.08.77	1.394,99-	20,91-	151,14-	1.525,22-
202000	200001	2.08.77	36,00	3,24	3,60	36,36
202600	200002	2.08.77	3.655,00	109,65	389,99	3.935,34
			*****12.866,02	******1.200,99	******1.283,17	*****12.948,20

Anhang

```
DECLARE ZZ WAWE RABATT MWST BRUTTO.
WHEN START
     TYPE <PAGE 48>;
     PERFORM !KOPF.
WHEN FMT 2
     RELEASE AT END GOTO !ENDE.
WHEN FMT 1
     TYPE (3)¦LZ <DEFER>;
     RELEASE AT END GOTO !ENDE.
TYPE <SKIP 4> (2)¦LZ
<SKIP 13> (1)¦´0_.__.__´
<SKIP 24> (3)¦´_.___._0_,__-´
<SKIP 40> (4)¦´_.___._0_,__-´
<SKIP 56> (5)¦´_.___._0_,__-´
<SKIP 73> (6)¦´_.___._0_,__-´.
ADD (3) TO WAWE.
ADD (4) TO RABATT.
ADD (5) TO MWST.
ADD (6) TO BRUTTO.
ADD 1 TO ZZ.
IF ZZ > 44
    PERFORM !KOPF.
RELEASE AT END GOTO !ENDE.
!ENDE
TYPE <LF>
<SKIP 28> WAWE¦´___.___._*_,__-´
<SKIP 44> RABATT¦´___.___._*_,__-´
<SKIP 60> MWST¦´___.___._*_,__-´
<SKIP 77> BRUTTO¦´___.___._*_,__-´.
STOP.
!KOPF ENTER
TYPE <TOP> <LF> <LF>
´R E C H N U N G S - A U S G A N G S B U
 C H´ <LF>.
TYPE 91´-´.
TYPE ´Kto.Nr.  Re.-Nr.´
<SKIP 22> ´Datum´
<SKIP 33> ´Warenwert´
<SKIP 52> ´Rabatt´
<SKIP 70> ´MwSt´
<SKIP 85> ´Brutto´.
TYPE 91´-´ <LF>.
MOVE 9 TO ZZ.
EXIT.
```