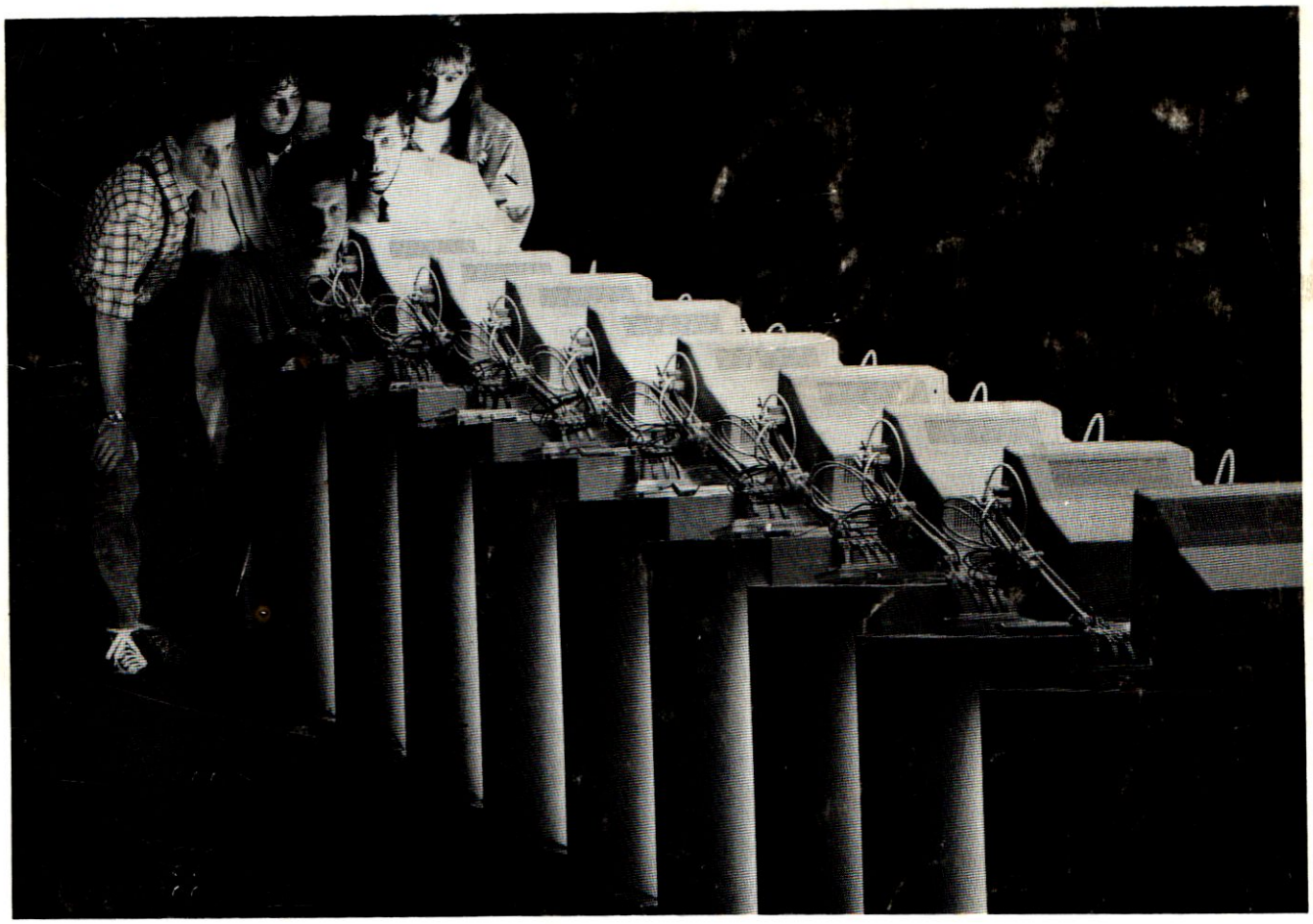
EDV-Dokumentation

NCR

ITX Betriebssystem
D00-0071251



Band 2

INHALT

Im Band 1 finden Sie die **Kapitel 1 bis 19,**
den Anhang und das Stichwort-Verzeichnis.

Im Band 2 finden Sie die **Kapitel ab 20,**
den Anhang und das Stichwort-Verzeichnis.

ITX Betriebssystem, Band 1:

		Kapitel/Seite
1.	Die ITX-System-Software	1/1
2.	Das ITX Multiprogramming-Betriebssystem	2/1
	2.1 Speicher-Belegung	2/1
	2.2 Prozesse	2/4
	2.3 Zeitliche Bedienung der Prozesse	2/5
	2.4 Zuteilung der Bildschirme	2/6
	2.5 Zuteilung der übrigen Peripherie	2/7
	2.6 Datei-Spezifikationen	2/12
	2.7 Anschluss an Netzwerke	2/21
3.	Bildschirm-Bedienung	3/1
	3.1 Tasten mit besonderer Funktion	3/1
	3.2 Bildschirm-Aufteilung	3/4
	3.3 Der Bildschirm-Dialog	3/5
	3.4 Meldungen vom Betriebssystem	3/6
	3.5 Abmelden / Ausschalten des Bildschirms	3/6
	3.6 Automatische Bildschirm-Abmeldung	3/7
	3.7 PC-Terminals mit ITX-WINDOWS	3/7
4.	Die System-Befehle	4/1
	4.1 Uebersicht	4/1
	4.2 System-Befehle (alphabetische Folge)	
	ABORT	4/ABO/1
	ALTER siehe Kapitel 5 und 20	
	ASSIGN	4/ASS/1
	ATTACH	4/ATT/1
	BEGIN	4/BEG/1
	BYE	4/BYE/1
	CHANGE	4/CHA/1
	CHECK	4/CHE/1
	COPY	4/COP/1
	DEASSIGN	4/DEA/1
	DELETE	4/DEL/1
	DETACH	4/DET/1

Band 1: System-Befehle (Fortsetzung) **Kapitel/Seite**

DISPLAY		4/DIS/1
DUMP		4/DUM/1
EDIT	siehe Kapitel 12	
END		4/END/1
EQUATE	siehe Kapitel 21	
ESCAPE	siehe Kapitel 11	
EXECUTE		4/EXE/1
EXIT	siehe Kapitel 21	
FIX		4/FIX/1
GO TO	siehe Kapitel 21	
GROUP	siehe Kapitel 22	
HELP		4/HEL/1
IF	siehe Kapitel 21	
LOAD	siehe Kapitel 23	
MOUNT		4/MOU/1
MOVE		4/MOV/1
NETEXECUTE		4/NET/1
NETLOGON		4/NET/3
QBACKUP		4/QBA/1
QRESTORE		4/QRE/1
REMOVE		4/REM/1
RESUME		4/RES/1
RETURN		4/RET/1
SET		4/SET/1
STOP	siehe Kapitel 11	
SUBMIT	siehe Kapitel 20	
SUSPEND		4/SUS/1
TIME		4/TIM/1
UNLOAD	siehe Kapitel 23	
WHEN	siehe Kapitel 21	
VIEW		4/VIE/1

5. Auto-Spooling	5/1
Kurzbeschreibung	5/1
ABORT Auto-Spooling	5/ABO/1
ALTER Auto-Spooling	5/ALT/1
ASSIGN Auto-Spooling	5/ASS/1
ATTACH Auto-Spooling	5/ATT/1
DETACH Auto-Spooling	5/DET/1
DISPLAY SPOOL	5/DIS/1
RESUME Auto-Spooling	5/RES/1
SUSPEND Auto-Spooling	5/SUS/1

Band 1	**Kapitel/Seite**
6. Das Streamer-Band-Utility ($STREAM)	6/1
Starten von $STREAM	6/2
BACKUP	6/3
BUILD	6/6
DISPLAY	6/8
DUMP	6/10
ERASE	6/12
INITIALIZE	6/13
RESTORE	6/15
RETENSION	6/19
VERIFY	6/20
Beispiel eines Laufes von $STREAM	6/21
7. Magnetband-Initialisierung ($MINT)	7/1
8. Disc-Initialisierung ($DINT)	8/1
8.1 Funktionen	8/1
8.2 Disc-Formate und Disc-Einteilung	8/4
8.3 Durchführung von $DINT	8/7
8.4 Beispiele	8/17
9. Der Sort ($Sort)	9/1
9.1 Möglichkeiten	9/1
9.2 Durchführung einer Sortierung	9/4
9.3 Die Sort-Parameter	9/8
9.4 Sort-Beispiele	9/15
10. Das Programm $HELP	10/1
11. System-Start, -Abschluss und Abbrüche	11/1
11.1 System-Start	11/1
11.2 System-Start Modell 10000/85	11/8
11.3 System-Abschluss	11/11
11.4 STOP	11/14
11.5 Arbeits-Abbrüche	11/15
11.6 ESCAPE	11/17
12. Der SCL-EDITOR EDIT	12/1
12.1 Möglichkeiten	12/1
12.2 Das Wichtigste über Start und Befehle	12/2
12.3 Befehls-Auswahl	12/4
12.4 Der Programmstart (Befehl EDIT)	12/9
12.5 Das Editor-Workfile	12/13
12.6 Die Editor-Befehle, allgemeines	12/16

Band 1: SCL-Editor-Befehle (Fortsetzung) **Kapitel/Seite**

1 A (ASCII)	12/1X/1
1 R (Replace)	12/1X/2
Alter	12/A/1
Copy	12/C/1
Display	12/D/1
Erase	12/E/1
Find	12/F/1
Grid	12/G/1
Help	12/H/1
Insert und ^ bzw. 1I	12/I/1
Justify	12/J/1
Mark	12/M/1
Numbers	12/N/1
Option	12/O/1
Page	12/P/1
Quit	12/Q/1
Repeat	12/R/1
Substitute	12/S/1
Transfer	12/T/1
Update	12/U/1
View	12/V/1
Window	12/W/1
<RET> oder Pfeil-ab	12/ZUS/1
Backspace oder Pfeil-auf	12/ZUS/1
Space (Leertaste)	12/ZUS/1
?	12/ZUS/2
< und >	12/ZUS/3
*	12/ZUS/3

12.7 Editorläufe ab Controlstring 12/ZUS/4

Anhang Anhang/1

ASCII-Code für ITX	Anhang/1
ASCII-Code für PC	Anhang/2
Umrechnung Dezimal/binär/hex	Anhang/3
EBCDIC-Code	Anhang/4
File-Status-Werte COBOL 74	Anhang/5
File-Status-Werte COBOL 85	Anhang/7

Alphabetisches Stichwort-Verzeichnis Stichw/1

ITX Betriebssystem, Band 2:

	Kapitel/Seite
20. Background-Verarbeitungen (Submit-Batch)	20/1
Uebersicht	20/1
ABORT Background-Job	20/ABO/1
ALTER Background-Job	20/ALT/1
DISPLAY Background-Jobs	20/DIS/1
SET Background-Jobs	20/SET/1
SUBMIT	20/SUB/1
21. Control-Strings	21/1
21.1 Uebersicht	21/1
21.2 Variablen	21/4
21.3 System-Befehle für Controlstrings	21/8
DISPLAY JCL	21/DIS/1
DISPLAY 'Meldung' mit W	21/DIS/2
EQUATE	21/EQU/1
EXECUTE Controlstring	21/EXE/1
Substitutions-Parameter	21/EXE/5
EXIT	21/EXI/1
GO TO	21/GO/1
IF	21/IF/1
LIST	21/LIS/1
NOABORT	21/NOA/1
SET JCL	21/SET/1
WHEN	21/WHE/1
22. Multi-Section/Multi-Volume-Files	22/1
22.1 Möglichkeiten	22/1
22.2 Erforderliche System-Befehle	22/1
GROUP und DISPLAY GROUP	22/4
ASSIGN (Multi-Section-Dateien)	22/9
DELETE (Multi-Section-Dateien)	22/12
DISPLAY Directory (Multi-Section-Dateien)	22/13
COPY, MOVE, QBACKUP (Multi-Section-Dateien)	22/14

Band 2 **Kapitel/Seite**

23. System-Optimierung 23/1
23.1 Uebersicht 23/1
23.2 Kontrolle der System-Auslastung 23/2
DISPLAY STATUS USAGE 23/3
EX $PDP 23/6
Snapshot 23/7
Data Collection und Data Analysis 23/11
23.3 Prozess-Prioriäten 23/18
SET PRIORITY 23/19
23.4 Vorausladen von Programmen 23/20
LOAD 23/20
UNLOAD 23/22
DISPLAY PROGRAMS 23/23
23.5 Load Leveling 23/24
SET USAGE 23/25
23.6 Disc-Cache 23/27

24. Start und Abschluss von TAM 24/1

25. Das Hardware-Fehler-Protokoll (Error-Log, $LOGUTIL) 25/1

26. Unterhalt des Security-Systems ($ACCESS) 26/1
26.1 Umfang des Zugriffs-Schutzes 26/1
26.2 Konzept des Programms $ACCESS 26/3
26.3 Erstmaliges Generieren des Security-Systems 26/6
26.4 Beschreibung der Bilder (Administrator-User) 26/8
26.5 Beschreibung der Bilder (Normal-User) 26/28
26.6 Befehls-Modus für $ACCESS 26/31

27. Konversion von Objektprogrammen($OBJCONV) 27/1

28. System-Disc rekonstruieren 28/1

Band 2 **Kapitel/Seite**

29. Der Text-Editor ($EDIT) 29/1
29.1 Möglichkeiten 29/1
29.2 Das Editor-Workfile (EWF) 29/3
29.3 Starten des Text-Editors 29/6
29.4 Uebersicht über die Text-Editor-Befehle 29/8
29.5 Beispiele von Text-Editor-Läufen 29/11
29.6 Text-Editor-Befehle 29/14
ADJUST 29/AD/1
AGAIN 29/AG/1
ASSIGN 29/AS/1
CHANGE 29/CH/1
CONCAT 29/CO/1
DCL 29/DC/1
DELETE 29/DE/1
DISPLAY 29/DI/1
ERASE 29/ER/1
FIND 29/FI/1
GRID 29/GR/1
INSERT 29/IN/1
MOVE 29/MO/1
POSITION 29/PO/1
PRINT 29/PR/1
QUIT 29/QU/1
RESEQUENCE 29/RE/1
SAVE 29/SA/1
SEARCH 29/SE/1
STRING 29/ST/1
TAB 29/TA/1

30. Die COBOL-Compiler ($COBOL, $COBOL9, COBOL85)
30.1 Uebersicht 30/1
30.2 Durchführung einer Compilation 30/5
30.3 Struktur des Object-Programms 30/16

31. Speicher-Formate auf magnetischen Datenträgern 31/1
31.1 Disc 31/1
31.2 Magnetband 31/10
31.3 Stremaer-Band und Helical Scan Tape 31/13

32. NCR Fern-Unterstützung 32/1

Band 2	**Kapitel/Seite**
Anhang	Anhang/1
ASCII-Code für ITX	Anhang/1
ASCII-Code für PC	Anhang/2
Umrechnung Dezimal/binär/hex	Anhang/3
EBCDIC-Code	Anhang/4
File-Status-Werte COBOL 74	Anhang/5
File-Status-Werte COBOL 85	Anhang/7
Alphabetisches Stichwort-Verzeichnis	Stichw/1

Bemerkungen zur Seiten-Numerierung:

Die Seiten sind alpha-numerisch-aufsteigend numeriert, unabhängig von der Länge der Seiten-Nummer.

Deshalb steht beispielsweise
die Seite 4/3
vor der Seite 4/MOU/1 und diese wiederum
vor der Seite 4/MOU/3 und diese wiederum
vor der Seite 20/DIS/1

Im Band 1 finden Sie die **Kapitel 1 bis 19,**
den Anhang und das Stichwort-Verzeichnis.

Im Band 2 finden Sie die **Kapitel ab 20,**
den Anhang und das Stichwort-Verzeichnis.

20. BACKGROUND-VERARBEITUNGEN (SUBMIT-BATCH-VERARBEITUNGEN)

UEBERSICHT:

Batch-Prozesse sind Verarbeitungen, die während ihres Ablaufes keinen Bildschirm belegen. Sie werden zwar an einem Bildschirm gestartet, doch wird dieser Schirm sofort nach dem Start-Befehl für andere Arbeiten frei. SUBMIT-Batch-Jobs laufen "im Hintergrund" des Systems ab.

- Für jeden SUBMIT-Batch-Job muss ein Control-String mit den entsprechenden System-Befehlen vorbereitet sein.

- Dieser Controlstring wird mit dem System-Befehl SUBMIT (anstelle von EX) aufgerufen.

- Im Normalfall ergibt jeder SUBMIT-Befehl einen Batch-Job, der einen Job-Namen erhält. Ohne besondere Angabe heisst dieser Jobname JOBNnnnnn, wobei die Nummer nnnnn vom System gebildet wird.

  Solche Jobs werden gemäss ihrer Priorität in Warteschlangen gespeichert und vom System automatisch ausgeführt, sobald genügend Systemkapazität frei ist.
  Es gibt maximal 8 Queues (Warteschlangen), je eine pro Priorität.
  Bei Bedarf lassen sich Batch-Jobs bilden, die vorerst im Hold-Zustand (Wartezustand) verbleiben, bis sie von der Bedienung zur Ausführung freigegeben werden.
  Falls erforderlich, können Batch-Prozesse auch direkt ausgeführt werden, ohne dass sie zuerst in eine Warteschlange kommen.

- Während der Ausführung eines SUBMIT-Batch Jobs bildet dieser einen eigenen Prozess, der im System-Status als Batch-Prozess erscheint. Sobald der Controlstring fertig abgelaufen ist, verschwindet der BATCH-Prozess aus dem System.

- Alle Ausgaben des Batch-Prozesses, die bei interaktiver Ausführung auf den Bildschirm ausgegeben würden, schreibt das System auf ein Protokoll-File. Dieses sollte nach Ablauf des Prozesses kontrolliert und gelöscht werden. Name, Disc-Station und Grösse für das Protokoll-File können im SUBMIT-Befehl angegeben werden.

VORBEREITUNG DES CONTROL-STRINGS

- Alle im SUBMIT-Batch-Prozess benötigten System-Befehle (z.B. ASSIGN, SET SW, EXECUTE usw.) müssen in diesem Control-String enthalten sein.

- Ein SUBMIT-Batch-fähiger Control-String darf nur System-Befehle des "SUBMIT-Befehls-Set" enthalten:

ASSIGN	ABORT
ALTER	ATTACH
BEGIN	CHECK
COPY	DELETE
DETACH	DISPLAY
DUMP	END
EQUATE	EXECUTE
EXIT	FIX
GROUP	HELP
IF	LINK
LOAD	MOVE
MOUNT	QBACKUP
QRESTORE	REMOVE
RESUME	RETURN
SET	SUBMIT
SUSPEND	TIME

Ein SUBMIT-Batch-Job kann auch seinen Start-Bildschirm mit ATTACH und DETACH ansprechen.

Daten-Eingaben

Der Control-String kann Programme mit Bildschirm-Eingaben (COBOL-Befehle ACCEPT) enthalten:
Unmittelbar nach dem EX-Befehl für ein solches Programm ist für jeden durchlaufenen ACCEPT-Befehl eine Zeile mit der entsprechenden Eingabe in den String einzuschliessen. Die Eingabedaten sind in diesen Zeilen ganz links zu speichern; bei numerischen Daten sind Vornullen erlaubt.
Die Anzahl solcher Eingabe-Zeilen muss genau mit der Anzahl durchlaufender ACCEPT-Befehle des Programms übereinstimmen.

Vorsicht:
Enthält der Control-String zu wenig Eingabezeilen, bleibt der Background-Job im I/O-Wait-Status stehen und muss mit ABORT J=jobname abgebrochen werden.
Enthält er zu viele Eingabezeilen, werden die überzähligen nach dem Ablauf des betreffenden Programmes als nachfolgende System-Befehle interpretiert.

SUBMIT-Batch-Verarbeitungen dürfen somit keine dynamischen Bildschirm-Dialoge enthalten: Die Anzahl Bildschirm-Eingaben (ACCEPT-Befehle im Programm) müssen fix und ihre Reihenfolge von vorangehenden Eingaben unabhängig sein.

Beispiel:

Ein Programm verlangt eine Disc-Datei, einen Drucker und folgende 3 Bildschirm-Eingaben:

- eine 6-stellige Zahl
- einen 10-stelligen Text
- einen 1-stelligen Code

```
AS DI physname (1)
AS LO (LP)
EX programm (n)       ;  Programm-Eingaben:
300980                ;  6-stellige Zahl
TEXTWORT              ;  10-stelliger Text
A                     ;  1-stelliger Code
```

Hinweis:
Wie Daten-Zeilen durch Controlstring-Variablen ersetzt werden können, finden Sie am Ende dieses Kapitels "Submit-Batch-Verarbeitungen".

AUSFUEHRUNG EINES SUBMIT-BATCH-PROZESSES:

Mit dem System-Befehl SUBMIT. Siehe weiter hinten im diesem Kapitel.

DAS SUBMIT-BATCH-PROTOKOLL-FILE

Jeder SUBMIT-Batch-Prozess erstellt ein solches File.

Es enthält:

- Prozess-Identifikation, Start-Datum und -Zeit (gleicher Text wie beim Start eines neuen Bildschirm-Prozesses)
- den Namen des Control-Strings und alle durchlaufenen System-Befehle.
- alle durch Programme mit COBOL-DISPLAY-Befehlen ausgegebenen Meldungen.
  Dabei ergeben die COBOL-Klauseln

LINE n	=	einen Zeilenvorschub
POSITION n	=	Druck-Position n
ERASE	=	neue Zeile

- gegebenenfalls Fehlermeldungen, die sonst auf dem Bildschirm erscheinen würden (z.B. File-Status-Meldungen, Fehler in Control-String-Befehlen usw.)
- eine Abschluss-Meldung:

  CONTROL-STRING physname DONE = normales Ende
  oder
  CONTROL-STRING physname ABORTED AT LINE n = Abbruch

Meldungen, die mit DI '....' an einen Bildschirm gesendet werden:

- mit Bildschirmangabe (n): Erscheinen am betreffenden Bildschirm.
- mit Bildschirmangabe H (ohne W): Werden ins Protokoll-File geschrieben.
- mit Bildschirmangabe H und W: Erscheinen an der Konsole (meistens Bildschirm 0).

Meldungen über nicht betriebsbereite Geräte (ATTENTION REQUIRED...) erscheinen an derKonsole (meistens Bildschirm 0).

Datei-Spezifikationen für das Protokoll-File:

- Es ist ein Spoolfile und kann mit MOVE ausgedruckt werden.
- Es wird vom System automatisch erstellt.

  Name : Jobname oder Name aus dem SPO-Parameter.

  Disc : System-Disc (SYS3) oder (n) aus dem SPO-Parameter.

  Grösse : 100, AP oder Grösse aus dem L-Parameter

ARBEITEN NACH ABSCHLUSS DES SUBMIT-BATCH-PROZESSES

- Ausgabe des Protokoll-Files zur Kontrolle des Ablaufes durch
  Ausdrucken auf einen Drucker oder Ausgabe auf den Bildschirm mit
  VIEW, COPY oder MOVE.
- Löschen des Protokoll-Files
  mit dem System-Befehl DEL.

SYSTEM-BEFEHLE IM ZUSAMMENHANG MIT BACKGROUND-VERARBEITUNGEN

Die folgenden System-Befehle haben mit Background-Verarbeitungen zu tun:

ABORT J=jobname	Löschen eines Background-Jobs.
ALTER J=jobname	Aendern von Job-Spezifikationen.
DISPLAY BATCH	Kontrolle der auf die Ausführung wartenden Background-Jobs.
DISPLAY STATUS	Kontrolle der laufenden und der abgelaufenen Background-Jobs.
SET BATCH	Submit-Batch-Prozess-Starte erlauben oder stoppen.
SUBMIT	Bilden und Starten eines Background-Jobs.

Diese Befehle werden anschliessend in alphabetischer Reihenfolge beschrieben.

Am Ende des Kapitels folgen einige Beispiele zur Background-Verarbeitung als Ganzes.

ABORT (Background-Jobs)

Funktion:

Löschen von Background-Jobs.

Format:

ABORT J=jobname

Beschreibung:

Der SUBMIT-Batch-Job mit dem jobnamen wird sofort aus dem System gelöscht.
Dieser Befehl ist auch während des Ablaufes eines Jobs möglich.

Siehe auch unter dem Befehl SUBMIT

Bemerkungen:

Dieser Befehl kann auch für abgelaufene SUBMIT-Batch-Jobs verwendet werden, um jene Jobs aus dem System-Status zu löschen. Anschliessend kann der gelöschte Jobname wieder neu vergeben werden.

Die Löschung eines SUBMIT-Batch-Jobs verursacht die folgende Meldung:

```
... JOB  jobname  KILLED
```

Beispiel

Löschen des SUBMIT-Batch-Jobs JOBN000002:

```
ABORT  J=JOBN000002
```

- Notizen -

ALTER (Background-Jobs)

Funktion:

Aendern von Background-Job-Spezifikationen.

Format:

```
ALTER J=jobname  /  I     \  [ RO=/Y\ ]
                 | Q=q     |       \N/
                 | HOLD    |
                 | RE      |
                 | F       |
                 | L       |
                 | UP=n    |
                 \ DOWN=n  /
```

jobname	Name eines SUBMIT-Batch-Jobs im System
I	Der Job soll sofort ausgeführt werden, unabhängig von der Queue, in welcher er sich befindet.
Q=q	q bezeichnet die Queue (Warteschlange), wo der Job neu einzureihen ist. Er wird zuhinterst in die Queue eingereiht.
HOLD	Der Job wird in den "HOLD"-Zustand versetzt. Er wird nicht gestartet, bevor er durch einen weiteren ALTER-Befehl mit dem RE-Parameter zur Ausführung freigegeben wird.
RE	Ein Job im "Hold"-Zustand wird zur Ausführung freigegeben. Allfällige DATE oder Zeit-Parameter aus dem SUBMIT-Befehl werden damit übersteuert. Der Job wird in die Queue eingereiht, die im SUBMIT- bzw. im letzten ALTER-Befehl für diesen Job ausgegeben wurde.

UP=n DOWN=n	Der Job wird innerhalb der laufenden Queue um n Positionen vorgeschoben (UP) bzw. zurückgesetzt (DOWN).
F	Der Job wird zuvorderst in die laufende Queue plaziert.
L	Der Job wird zuhinterst in die laufende Queue plaziert.
RO=Y	Der Job wird automatisch vorübergehend aus dem Memory ausgelagert (rolled out), falls Memory-Knappheit herrscht (stillschweigende Annahme).
RO=N	Der Job wird auch bei Memory-Knappheit nicht "rolled out".

Bemerkungen:

Dieser Befehl ist nur für SUBMIT-Batch-Jobs verwendbar, welche mit SUBMIT ohne den I-Parameter gebildet wurden und sich noch in einer Batch-Warteschlange (Queue) befinden.

Beispiele:

a) Den Job JOB01 zuvorderst in die Queue 4 einsetzen:

```
1. AL  J=JOB01  Q=4
2. AL  J=JOB01  F
```

b) Den Job JOBN0002 sofort starten

```
AL  J=JOBN0002  I
```

c) Den Job JOB03 in den Wartestatus (HOLD-Queue) versetzen

```
AL  J=JOB03  HOLD
```

d) Den in der HOLD-Queue wartenden Job zur Ausführung freigeben. Er soll nicht "rolled out" werden

```
AL  J=JOB03  RE  RO=N
```

- Notizen -

DISPLAY (Background-Jobs)

Funktionen:

Im Zusammenhang mit Background-Verarbeitungen bestehen zwei DISPLAY-Varianten:

1. Ausgabe des Batch-Queue-Verzeichnisses
   (DISPLAY BATCH)

2. Zusätzliche Angaben im System-Status
   (DISPLAY STATUS)

1. AUSGABE DES BATCH-QUEUE-VERZEICHNISSES

Ausgabe aller mit SUBMIT eingegebenen Batch-Jobs, die sich noch in Warteschlangen befinden.
In Ausführung begriffene oder schon beendete SUBMIT-Batch-Jobs sind im System-Status ersichtlich.

Formate:

DISPLAY BATCH [**Q=/q**] [TO (/ n \,/**LP** \ )]
\H/ \ANY/ **DLP**/

DISPLAY BATCH F

Varianten:

ohne Q=...	Ausgabe aller wartenden Batch-Jobs
Q=q	Nur die Jobs in der Warteschlange q, die nicht im HOLD-Status sind. (q = 1 bis 8)
Q=H	Nur die Jobs im Hold-Status.
mit F	Zeigt nur das Batch-Flag, auch wenn keine wartenden Jobs vorhanden sind. (vgl. Befehl SET BATCH)

Ausgabe von DISPLAY BATCH:

```
  BATCH QUEUES STATUS AT 89/07/28  19:08:03.90       (BATCH QUEUE FLAG = ON )

QUEUE POS  USER ID  JOB NAME  SYSIN FILE  STATUS   DATE    TIME   ROLL
   8               JOBN000009 C-NACHT      HOLD  89/07/28 22:00   YES
   3               MONEND     C-MONEND     HOLD                   YES
   8    1          JOBN000004 SU000AP150 PENDING                   YES
   8    2          JOBN000005 C-SUB200   PENDING                   YES
```

BATCH QUEUE FLAG	Siehe System-Befehl SET BATCH
QUEUE	Warteschlange 1 bis 8
POS	Position in der Warteschlange, 1 = zuvorderst
JOB NAME	Jobname
SYSIN FILE	Mit SUBMIT gestarteter Control-String
STATUS	HOLD = Wartet bis auf weiteres bzw. bis zur angegebenen DATE/TIME PENDING = Auf Ausführung wartend. Wird automatisch gestartet DESPOOL = Am Ablaufen DONE = Beendet
DATE/TIME	Im SUBMIT bestimmte früheste Ausführungszeit
ROLL	YES = Job wird bei Memory-Knappheit ausgelagert NO = Job wird nicht ausgelagert.

2. ANGABEN IM SYSTEM-STATUS FUER ABGELAUFENE BACKGROUND-JOBS

Die allgemeine Beschreibung des System-Befehls DI ST finden sie im Kapitel "System-Befehle".

Besondere Angaben für abgelaufene Background-Jobs

- **Im Teil P, Prozess-Verzeichnis:**

  Abgelaufene SUBMIT-Batch-Jobs bleiben während einiger Zeit nach dem Abschluss wie folgt im Status (am Ende des Prozess-Verzeichnisses):

```
JOB ID      CS FILE  PROGRAM  STATUS  MODE  CPUTIME

JOBN000001                    TERM          00:01:04.60
```

  CPUTIME = Prozessorzeit, welche der abgelaufene Batch-Job benötigte.

  Solche abgelaufene Jobs lassen sich mit ABORT J=jobname aus dem Status löschen.

- **Bei Angabe des L-Parameters im Befehl DI ST ALL:**

  z.B: DI ST ALL L

  Für jeden abgelaufenen SUBMIT-Batch-Job werden folgende Details ausgegeben:

```
1.38 JOBN000006                          SPOOL Q          000:00:00.77
   USER-ID =
   SYSIN =  SU001BK115/1      VSN = 350000
   SYSOUT =  SP001BK115/1      VSN = 350000
0.22 JOBN000004                            TERM           000:00:02.10
   USER-ID =
   SYSIN =  SU000AP150/1      VSN = 350004
   SYSOUT =  SP000AP150/1      VSN = 350004
```

SYSIN	Name/Generation des Control-Strings
VSN	Packnr, wo der Control-String gespeichert war
SYSOUT	Name des Protokoll-Files (SPO=)
VSN	Packnr, wo sich das Protokoll-File befindet

Nur diese Angaben eines bestimmten Submit-Jobs erhält man mit dem Befehl

DI ST J=jobname L

SET (Background-Jobs)

Funktion:

Erlauben/Verbieten von Batch-Prozess-Starts.

Format:

```
SET BATCH  /ON \
           \OFF/
```

Beschreibung:

ON : Mit SUBMIT gebildete Batch-Jobs werden vom System automatisch gestartet, gemäss ihrer Priorität und entsprechend der System-Auslastung.

OFF : Das System startet keine SUBMIT-Batch-Prozesse mehr. Wartende Jobs bleiben in ihren Warteschlangen. Ihre Ausführung ist erst wieder möglich, wenn SE BA ON ausgeführt wird.
SUBMIT-Befehle mit dem I-Parameter werden nicht mehr akzeptiert.

Nach dem System-Start gilt stillschweigend SE BA ON.

Beispiele:

a) Den Start neuer SUBMIT-Batch-Prozesse stoppen:

```
SE BA OFF
```

b) Den Start neuer SUBMIT-Batch-Prozesse wieder erlauben:

```
SE BA ON
```

- Notizen -

SUBMIT (Background-Jobs)

Funktion:

Starten eines Background-Batch-Jobs (Submit-Batch-Job).

Format:

```
SUBMIT c-string (n,DI)  [V=packnr]  [J=jobname]
  [P=p]  [SPO=protokoll(n,DI)] [L=zeilen] [zeit]
  [NOR]  [JC=j]  [SW=/n [,n] ...\ ]
                        \ ALL     /
  [ /[Q=q] [DA=jjmmtt] / [HOLD=st:mi]\ ]
                       \   [HOLD]    /
  [substitute]
```

Parameter-Beschreibung

Allgemeine Bemerkungen:

- Die Reihenfolge der Parameter nach c-string (n DI) ist frei.
- Die meisten Parameter sind nur in besonderen Fällen oder nur erforderlich, wenn von den Standard-Annahmen abgewichen wird. Die Standard-Annahmen sind in den folgenden Erklärungen in Klammern mit dem Stichwort "Default": angegeben.

Parameter des SUBMIT-Befehls:

c-string (n, DI)	Name und Disc-Station des Control-String, der im Background ablaufen soll.
V=packnr	Packnummer, wo sich dieser Control-String befindet. Diese Angabe übersteuert die Disc-Station (n, DI) nach c-string. Falls beim Start der Job-Ausführung keine Disc-Einheit mit dieser Packnr mounted ist, verlangt das System am Schirm 0 eine solche Disc-Einheit mit der Meldung Q002 JOB jobname REQUIRES PACK, VOL-SER-NBR: packnr Antworten: - Disc bereitmachen und NEWLINE drücken: Der Job läuft. - X eintippen: Der Job wird abgebrochen und vergessen!
J=jobname	Name des Job, gemäss den Regeln für lognamen. (Default: JOBNnnnn, wobei nnnn eine vom System gebildete Nummer darstellt). Bemerkung: Solange ein Jobname in einer Warteschlange oder im Systemstatus vorhanden ist, kann derselbe Job-Name nicht neu vergeben werden!.
Q=q	q = Nummer der Warteschlange, 1 - 8. Die Jobs in Queue 1 werden als erste gestartet, diejenigen in Queue 8 zuletzt. (Default: = 8).

HOLD	Der Job wird im Hold-Zustand gespeichert. Er wird nicht ausgeführt, bevor er mit dem Befehl ALTER J=jobname RE aktiviert wird.
HOLD=st:mi	st:mi = Zeit als Stunde : Minute, wann der Job in die Warteschlange mit der tiefsten Priorität eingereiht und zur automatischen Ausführung freigegeben wird. Ohne DA=-Parameter darf die Zeit nicht vor der aktuellen Uhrzeit liegen. (Default: sofort).
DA=jjmmtt	jjmmtt = Datum, 6-stellig, als Jahr, Monat und Tag, an dem der Job in die Warteschlange eingereiht und zur automatischen Ausführung freigegeben wird.
I	Immediate. Der Job wird sofort als Background-Prozess gestartet. Er wird in keine Warteschlange eingereiht. Die Variante gehört dem privilegierten Befehlssatz an.
P=p	p = Ausführungs-Priorität des Background-Prozesses (Default = 8).

SPO=protokoll (/n DI \ )
|n **LP** |
\n **DLP**/

protokoll = Name des Protokoll-Files und Disc-Station bzw. Drucker.
Das Protokoll-File wird als neues File eröffnet.
(Default: Jobname auf Disc SYS3, höchste vorhandene Generation + 1. Wird keine Disc-Station angegeben, erstellt das System das Protokoll auf Disc SYS3).

L=zeilen	zeilen = Anzahl Zeilen, die das Protokoll-File umfassen soll, maximal 999999. (Default: 300 Zeilen = 100 Sektoren).
zeit	Maximale Prozessor Zeit für diesen Prozess, in Sekunden. Ist diese CPU-Time aufgebraucht, wird der Prozess abgebrochen. Maximum: 2,147,483. (Default: 1,000,000 = ca. 24 Stunden).
NOR	NO ROLL: Dieser Job soll auch bei Memory-Knappheit nicht aus dem Memory ausgelagert werden, auch wenn der "High level" erreicht ist. - Siehe unter "Load-leveling".
JC=j	j = Anfangswert des JCL-Code für den Batch-Prozess, 0 - 7. (Default = 0).
SW=/n [n]...\ \ ALL /	Anfangsstellung der SWITCHES für den Batch-Prozess. n = Switch-Nummern, die ON gesetzt werden. ALL = Alle Switches ON setzen. (Default: Alle Switches OFF)
substitute	Siehe im Kapitel "Control-Strings". Die maximale Länge aller Substitutionsangaben beträgt 256 Zeichen.

Wichtige Bemerkung:

Damit der Namen des Protokoll-Files eindeutig und voraussagbar ist, muss entweder der Parameter J=jobname oder SPO=protokoll (n) oder beides angegeben werden.

Befehlssatz-Zugehörigkeit:

Sofern dem Schirm SUBMIT-Batch-Prozesse erlaubt sind:

- ohne den I-Parameter : privilegiert und normal.
- mit dem I-Parameter : privilegiert.

SUBMIT-Befehle in einem interaktiven Control-String

Ein interaktiver Control-String, der mit EXECUTE gestartet wird, kann SUBMIT-Befehle enthalten. Dabei ist jedoch zu beachten, dass nach dem Start solcher Batch-Prozesse der aufrufende Control-String parallel zum Batch-Prozess fortgesetzt wird. Die SUBMIT-Batch-Prozesse laufen unabhängig zuende ohne Rückkehr in den Aufruf-Control-String.

MELDUNGEN VON SUBMIT-BATCH-JOBS

- Am Schirm, an dem der SUBMIT-Befehl ausgeführt wird:

  - Unmittelbar nach der Eingabe des SUBMIT-Befehls:

    - falls SUBMIT ohne I-Parameter:
      ... JOB jobname SUBMITTED ON /QUEUE n \
      \HOLD QUEUE/

    - falls SUBMIT mit I-Parameter:
      ... PROCESS xx.yy CREATED

  - Wenn der Batch-Job fertig ausgeführt ist:

    X001 PROCESS xx.yy TERMINATED - xxxxxxxxx

    Der Prozess xx.yy wird damit aus dem System entfernt. Anstelle von xxxxxxxxx kann stehen:

NORMALLY	=	Prozess ohne System-Fehler beendet. Wichtig: Der Ablauf des Control-String muss durch Kontrolle des Protokoll-Files überprüft werden, da gewisse Fehlerbedingungen nur dort ersichtlich sind.
anderer Text	=	Verarbeitung nicht ordnungsgemäss ausgeführt. (siehe Meldungs-Verzeichnis).

- Am Operator-Bildschirm (in der Regel Schirm 0):

  - Beim Start der Ausführung des Job:

    Q000 JOB jobname INITIATED PROCESS-ID xx.yy

    Damit wird der Prozess xx.yy neu gebildet.

  - Wenn der SUBMIT-Batch-Job fertig ausgeführt ist:

    Q001 JOB jobname TERMINATED.

  Der Job und der dazugehörige Prozess verschwinden aus dem System. Während einiger Zeit lassen sich die Job-Einzelheiten mit DI ST ALL L kontrollieren.

BEISPIELE

a) Ausführung des Control-String C-1 von Disc-Einheit 1 mit dem Protokoll-File SP-C1 auf Disc 1, für maximal 200 Zeilen. Der Control-String wird zuerst in eine Warteschlange eingereiht.

```
SUB C-1 (1) SPO=SP-C1 (1) L=200
```

Kontrolle des Protokoll-Files am Bildschirm, nach Beendigung des Batch-Prozesses:

```
VI SP-C1 (1)
```

Löschen des Protokoll-Files:

```
DEL SP-C1 (1)
```

b) Sofort-Ausführung desselben Job, ohne ihn zuerst in eine Warteschlange einzureihen
(Vorsicht: kann andere Prozesse bremsen!).

```
SUB C-1 (1) SPO=SP-C1 (1) L=200 I
```

C) Ausführung des Control-String C-2 von Disc 2 mit allen Default-Annahmen.
Annahme: Das System nennt den Job JOBN0003, System-Disc SYS3 ist Station 0.

```
SUB C-2 (2)
```

Kontrolle des Protokoll-Files

```
   durch Ausdrucken:
COP JOBN0003/(SYS3) TO (LP)   neueste Generation!
   oder nur am Bildschirm:
VI JOBN0003/(SYS3)            neueste Generation!
```

Löschen des Protokoll-Files:

```
DEL JOBN0003/(SYS3)        neueste Generation!
```

Bemerkungen zu diesem Beispiel:
- Das Protokoll-File ist möglicherweise zu gross (100 Sektoren)
- Falls der Operator die Meldung mit dem Jobnamen übersieht, muss der Name des Protokoll-Files erraten werden!
- Da der Name des Protokoll-Files nicht voraussagbar ist, können Ausdrucken und Löschen desselben nicht in einem Control-String ausgeführt werden!

d) Ausführen des Control-String C-3 im Background, von Disc 1. Priorität = 3, Protokoll-File = PROT3 auf Disc 2, mit maximal 500 Zeilen, Processorzeit-Limite = 1800 Sekunden.

```
SUB C-3 (1) P=3 1800 SPO=PROT3 (2) L=500
```

e) Speichern des Batch-Job JOB12UHR, der erst um 12 Uhr zur Ausführung freizugeben ist. Bis dahin ist er in Queue 4 zu speichern. Der auszuführende Controlstring ist C-12 auf Disc 4.

```
SUB C-12 (4) J=JOB12UHR  HOLD=12:00  Q=4
```

Bemerkung:
Das Protokoll-File erhält den physnamen JOB12UHR und wird auf dem System-Disc SYS3 gespeichert.

ERSETZEN VON SUBMIT-DATENZEILEN DURCH CONTROLSTRING-VARIABLEN

Prinzip:

Daten-Zeilen in SUBMIT-Controlstrings werden nicht automatisch durch Controlstring-Variablen ersetzt. (vgl. auch Kapitel "Control-Strings")

Gleichwohl lassen sich solche Datenzeilen mit der folgenden, etwas aufwendigeren Technik durch Variablen ersetzen. Dabei wird der Editor benötigt, um die Daten-Zeilen im Submit-Controlstring durch die Variable(n) zu ersetzen.

Die Technik benötigt:

- Einen Controlstring, in dem die Variablen gebildet werden.
  Im Beispiel: C-WAHL

- Den Submit-Controlstring mit dem Start der Verarbeitung, mit AS-Befehlen, EX des Programms und Daten-Zeilen.
  Im Beispiel: C-SUB.
  Davon eine Arbeitskopie erstellen, in welcher die Daten-Zeilen ersetzt werden.
  Im Beispiel: C-SUB-ARB.

- Einen Editor-Controlstring, der die Daten-Zeilen im Submit-Controlstring (C-SUB-ARB) durch die Variablen ersetzt.
  Er enthält einen Substitute-Befehl pro Variable.
  Im Beispiel: C-ED.

Ein Beispiel finden Sie auf der nächsten Seite.

Beispiel:

Im Submit-Controlstring C-SUB wird eine Daten-Zeile mit der gewünschten Periode eingelesen. Diese Periode soll vor jedem Lauf am Bildschirm eingegeben werden können (Variable "PER"):

C-SUB: Arbeits-Controlstring, kopiert als C-SUB-ARB :

```
AS ....                 ; Assign der Datenfiles
AS ....                 ; Assign der Datenfiles
EX P-xxx(n)
PERIODE                 ; Daten-Zeile, wird ersetzt
```

C-WAHL: Bilden der Variable(n):

```
DEL C-SUB-ARB (n)           ; Arbeitskopie von C-SUB
COP C-SUB(n) C-SUB-ARB(n)   ; erstellen. C-SUB bleibt
                            ; unverändert
DI '   WAHL DER PERIODE   ' H
DI '   ----------------   ' H
DI ' PERIODE ?   ' H W=PER
; event. Plausibilitätstests ;
DEL SPO-ED (n)              ; Protokoll von SUB C-ED
SUB C-ED (n) SPO=SPO-ED(n) Ä #ALL Ü
```

C-ED: Ersetzt die Variable(n) in C-SUB-ARB :

```
ED C-SUB-ARB (n) CS ;EDITOR:
S                   ;ersetzt "PERIODE" durch die
PERIODEÖPERÖA E     ; Variable "PER" aus C-WAHL.
S                   ;falls Eingabe <RET> als
''Ö ÖA E            ; PER erlaubt ist.
Q
Y
;                   ;SUBMIT:
DEL SPO-SUB (n)     ; Protokoll von SUB C-SUB-ARB
SUB C-SUB-ARB(n) SPO=SPO-SUB(n)
```

Ausführung:

```
EX C-WAHL (n)
```

bewirkt, dass der Controlstring C-SUB-ARB wie folgt abläuft:

```
AS ....
AS ....
EX P-xxx(n)
PER                 ; Daten-Zeile = Variable PER
```

21. CONTROL-STRINGS

21.1 UEBERSICHT

Ein Controlstring ist eine Folge von System-Befehlen, die auf einem Datenträger (meist Disc) als Datei gespeichert ist.

Ausführung von Controlstrings

Zur Ausführung der ganzen Befehlsfolge ist ein EXECUTE- bzw. NETEX-Befehl (interaktiv) oder SUBMIT-Befehl (Batch) mit dem Namen des Controlstrings notwendig, z.B:

```
EX c-string (n)
```

Controlstrings, die als interaktive Prozesse ablaufen, dürfen alle Systembefehle enthalten.
In Controlstrings für Submit-Batch-Prozesse sind nur die Befehle des Submit-Befehlssatzes erlaubt.

Das Ende jedes abgelaufenden interaktiven Controlstrings wird auf dem dazugehörigen Bildschirm angezeigt mit der Meldung

```
CONTROL STRING Name DONE
                    (falls normal beendet)
oder:

CONTROL STRING Name ABORTED AT LINE nn
                    (falls abgebrochen).
```

Fehler in Controlstrings

Tritt während des Controlstring-Ablaufs ein Fehler auf, wird der String in den meisten Fällen mit einer entsprechenden Meldung abgebrochen.

Gewisse Fehler-Bedingungen verursachen jedoch keinen Controlstring-Abbruch bei Fehler, z.B:

Befehl	Fehler-Bedingung
AT (n,LP)	Drucker schon attached
AT (n,CT)	Bildschirm schon attached
CH physname(n) name	File "name" besteht schon
DEL physname(n)	File existiert nicht
DET (n,LP)	Drucker ist schon detached
DET (n,CT)	Bildschirm ist schon detached
DI physname(n)	File existiert nicht
DI (n)	Disc ist nicht mounted oder existiert nicht
MOU (n)	Disc ist schon mounted
REM (n)	Disc ist schon removed
WH	irgendeine Fehler-Bedingung

Mit dem Befehl NOABORT in einem Control-String kann ein Abbruch bei Fehlern verhindert werden.
Mit dem Befehl WH #ERROR unmittelbar nach einem Befehl lässt sich testen, ob ein Fehler aufgetreten ist.

Controlstrings mit verschiedenen Ablauf-Varianten

Das Betriebssystem kennt verschiedene Möglichkeiten der Varianten-Steuerung:

JCL-Code:
Jedem Prozess ist ein interner Code mit dem Namen JCL-Code zugeteilt.
Dieser kann von jedem Programm oder mit dem SET JCL-Befehl auf einen Wert von 0 bis 7 gesetzt werden.
Mit dem Befehl IF lassen sich vom Wert des JCL-Code abhängige Sprünge innerhalb eines Controlstring ausführen.

Substitution von Parametern:
Beim Start eines Controlstring lassen sich mit Substitutions-Parametern bestimmte Zeichenfolgen im Controlstring durch andere ersetzen.

Controlstring-Variablen:
Mit den Befehlen EQUATE und DISPLAY 'meldung' W=variable können im Controlstring variable Daten erzeugt werden.
Der Befehl WHEN erlaubt von diesen Daten abhängige Sprünge innerhalb des Controlstrings.

System-Abfragen:
Mit den Befehlen WHEN und EQUATE lassen sich systemliche Informationen testen oder in Variablen speichern, z.B: Datum, Zeit, Wochentag, Switch-Stellungen, JCL-Code, Prozess-ID, Test auf Existenz oder Grösse einer Datei usw.

Aufruf weiterer Controlstrings in einem Controlstring

In Controlstrings lassen sich mit EXECUTE beliebige weitere Controlstrings aufrufen.

Ohne besondere Angabe erfolgt nach der Ausführung eines weiteren aufgerufenen Controlstrings keine Rückkehr in den Aufruf-String.
Der LINK-Parameter im aufrufenden EXECUTE-Befehl jedoch bestimmt, dass die Steuerung am Ende des aufgerufenen Controlstrings wieder in den Aufruf-Controlstring zurückkehren soll.

Im EXECUTE-Befehl kann auch die Start-Zeile angegeben werden, wo die Ausführung beginnen soll.

Anzeige der durchlaufenen Controlstring-Befehle während des Ablaufes

Ohne besondere Angaben werden alle ausgeführten Befehle eines Controlstrings am Bildschirm angezeigt.

Mit dem Parameter NO im startenden EXECUTE-Befehl kann diese Anzeige unterdrückt werden.

Mit dem Befehl LIST lässt sich im Controlstring selbst festlegen, ob keine Befehle, nur die ausgeführten oder auch die durch logische Entscheide übersprungenen Befehle anzuzeigen sind.

21.2 VARIABLEN

Bildung von Variablen

Mit den Befehlen EQUATE und DISPLAY...W="variable" können im Dialog und in Controlstrings variable Daten erzeugt werden:

EQUATE **variable** inhalt

DISPLAY 'meldung' H W=**variable**

Solche Variablen können verwendet werden, um Filenamen, Geräte-Angaben, Befehls-Parameter, Meldungen usw. in Controlstrings von Bediener-Eingaben oder von logischen Entscheiden abhängig zu definieren.

Befehlscodes (AS, EX usw.), Sort-Parameter und Datenzeilen in Submit-Batch-Controlstrings lassen sich nicht direkt als Variable darstellen.
Um solche Angaben zu ersetzen, ist eine aufwendigere Technik erforderlich (vgl. am Ende des Kapitels "Submit-Batch-Verarbeitungen").

Beachten sie auch die Beispiele unter den Controlstring-Befehlen DISPLAY und EQUATE.

REGELN FUER VARIABLEN

Alphanumerische Variablen:
- Maximal 10-stellig
- Das 1. Zeichen muss ein Buchstabe sein.
- Darf nur aus Buchstaben, Ziffern und Bindestrichen bestehen, jedoch ohne Ä, Ö und Ü.

Beispiele:
FILE, A12, K04B, B-1

Numerische Variablen:
- Eine maximal 10-stellige Zahl, eventuell mit vorangehendem Minuszeichen.
- Keine Dezimalstellen.
- Vornullen werden ignoriert.

Beispiele:
1234, -15,
0034 ergibt 34 (Vornullen werden ignoriert!)

Meldungs-Variablen (NUR für Meldungen mit DI '.....')

Als Meldungs-Teile sind auch Variablen möglich, die nicht den obigen Regeln entsprechen. Solche Variablen sind zwischen Hochkommas zu setzen.

Beispiele:

'DAS FILE '
(Leerstellen !)

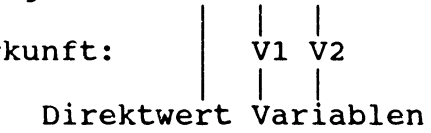
'MUTATION / LOESCHUNG '
(/, Leerstellen, über 10-stellig !)

Kombinations-Möglichkeit von Variablen:

Mit dem **Zeichen @** lassen sich mehrere Variablen und/oder Zeichenfolgen (direkte Werte) aneinanderreihen, z.B:

V1 habe den Inhalt D2
V2 habe den Inhalt 13

KOP@V1@V2 ergibt: **KOPD213**



Bemerkungen zur Kombination von Variablen:

- Die maximale Länge eines Ausdrucks von Variablen beträgt 10 Stellen (ohne die @-Zeichen).

Erweiterte Möglichkeiten für Ausdrücke, die nur für Meldungen mit DI 'meldung' bestimmt sind:

- Auch Variablen zwischen Hochkommas, die irgendeinen Inhalt aufweisen können, sind erlaubt.

- Die maximale Länge eines Ausdrucks ist nicht auf 10 Stellen beschränkt.

- Wenn in einem Ausdruck mit Display-Variablen (mit '....') der erste Teil nicht in Hochkommas steht, muss der Ausdruck mit
  ''@ (2 Hochkommas, dann @) beginnen,

  z.B:

```
V1 = FILE
V2 = ' A+B'

EQ AUSDR1 ''@V1@V2
          ergibt: 'FILE A+B'

EQ AUSDR2 ''@STAMM@V1@V2@' OK ?'
          ergibt: 'STAMMFILE A+B OK ?'

Jedoch:
EQ AUSDR V2@' OK ?'
          ergibt: ' A+B OK ?'
     Kein ''@ am Anfang,
     weil 1.Teil = Variable mit Hochkommas.
```

Weitere Beispiele finden Sie unter den Controlstring-Befehlen DISPLAY und EQUATE.

Beispiele für DISPLAY mit Variablen-Eingabe (Forts.)

b) Der folgende Control-String löscht den Bildschirm und baut ein Auswahlbild auf. Der Benutzer kann darauf zwei verschiedene Arbeiten wählen, die als Controlstrings abgespeichert sind.
Die Eingabe von <RET> beendet den Controlstring.

Vergleichen Sie auch die Befehle GO und WHEN in diesem Kapitel.

Variablen-Angaben sind **fett** gedruckt.

```
LIST OFF     ; Befehle am bilschirm nicht zeigen
BILD:
     ; Bild löschen und Titel ausgeben
DI '         AUSWAHL-BILD      ' H E
DI '         ------------      ' H
DI '         T = TAGES-ENDE    ' H
DI '         W = WOCHEN-ENDE   ' H
DI '         <RET> = ABSCHLUSS' H
     ; Abstand durch Ausgabe von Leerstellen
DI '  ' H
DI '  ' H
     ; Eingabe der Wahl
DI '    BITTE WAEHLEN SIE:    ' H  W=WAHL
          ;Test auf Eingabe = <RET>-Taste
WH WAHL = ''  GO TO ENDE
          ;Test auf Eingaben T oder W
WH WAHL = "T" GO TO TE
WH WAHL = "W" GO TO WE
     ; Ungültige Eingabe
GO TO BILD
     ; Eingabe = T:
     ; Aufruf des Controlstrings C-TAGEND
TE:
EX C-TAGEND (0) LINK
GO TO BILD
     ; Eingabe = W:
     ; Aufruf des Controlstrings C-WOCHEND
WE:
EX C-WOCHEND (0) LINK
GO TO BILD
          ;Ende des Controlstrings
ENDE:
```

Variable Meldungen

Mit der Technik zum Zusammensetzen von Variablen lassen sich variable Meldungen generieren.

Beispiel einer variablen Meldung:

Kontroll-Meldung nach Eingabe einer Variablen ausgeben:

a)

```
DI 'ZU REORGANISIERENDES FILE: ' H W=FIL
                        ; Annahme: Eingabe = D-KUNDE

EQ T1 'IST DER FILE-NAME ' ; Textvariablen, zwischen
EQ T2 ' OK ? (J/N) '       ; Hochkommas

DI T1@FIL@T2  H  W=ANTW
```

---> dieser DI -Befehl wird ausgeführt als:

```
DI 'IST DER FILE-NAME D-KUNDE OK? (J/N) ' H W="ANTW"
     |                |          |
Variablen: T1         FIL        T2
```

Wichtiger Hinweis:
Für weitere Varianten beachten Sie die Regeln für das Kombinieren mehrerer Variablen am Anfang des Kapitels "Control-Strings", Abschnitt "Variablen".

EQUATE

Funktion:

Bilden und Verändern von Variablen, vorwiegend in Control-Strings.

Format:

EQUATE **variable** [TO **ausdruck**]

Parameter:

variable — Name einer Variablen:
Ein Wort, 1- bis 10-stellig, aus Buchstaben, Ziffern, Bindestrichen. Das erste Zeichen muss ein Buchstabe sein.
Ueberall, wo dieser Name im laufenden Control-String bzw. in der laufenden Befehlsfolge vorkommt, wird das Resultat des "ausdruck" eingesetzt.

ausdruck — Ein Wert oder eine Formel zur Bestimmung des temporären Wertes der Variablen in der laufenden Befehlsfolge.
Der ausdruck kann auch SCL-Funktionen enthalten.
Beachten Sie auch den Abschnitt "Variablen" am Anfang des Kapitels "Control-Strings"!

ohne TO ausdruck — Die Variable mit dem Namen "variable" wird gelöscht.

Bemerkungen:

Jede ausgeführte EQUATE-Funktion wird mit der Meldung bestätigt:

... variable = wert

In der Befehlsfolge, wo der EQUATE-Befehl steht, werden alle Vorkommen der angegebenen Variable, die nach dem EQUATE-Befehl ablaufen, gemäss "ausdruck" ersetzt.

Kontrolle:
Der Wert aller Variablen der laufenden Befehlsfolge kann mit dem Befehl DI SUB sichtbar gemacht werden.

Einschränkung:
Befehlscodes (AS, EX usw.), Sort-Parameter und Datenzeilen in Submit-Batch-Controlstrings lassen sich nicht als Variable darstellen.

Uebergabe von Variablen:
Die Uebergabe von Variablen an andere Controlstrings ist am Anfang des Kapitels "Control-Strings" sowie unter dem Controlstring-Befehl EXECUTE beschrieben.

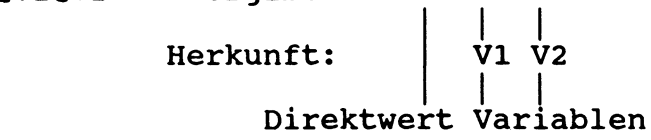
Vorsicht bei EQUATE-Befehlen im Dialog:
Wird ein EQUATE-Befehl ausserhalb eines Control-Strings ausgeführt, gilt der Ersatzwert für die angegebene Variable für die ganze Dauer des betreffenden interaktiven Dialogs.
Bei der Eingabe weiterer Befehle ist die Ersetzung des Variablennamens nicht sichtbar!

Bilden zusammengesetzter Variablen:

Mit dem **Zeichen @** lassen sich mehrere Variablen und/oder Zeichenfolgen (direkte Werte) aneinanderreihen, z.B:

V1 habe den Inhalt D2
V2 habe den Inhalt 13

KOP@V1@V2 ergibt: **KOPD213**



Wichtiger Hinweis:
Beachten Sie dazu den Abschnitt "Variablen" am Anfang des Kapitels "Control-Strings".

Beispiele:

a) Bilden einer Variablen FNAME mit dem Inhalt D-KUNDE:

Inhalt des Control-String:	läuft ab als:
EQ FNAME D-KUNDE	Meldung: ...FNAME=D-KUNDE
AS A **FNAME** (3)	AS A **D-KUNDE** (3)
EX P1 (10)	EX P1 (10)
COP **FNAME** (3) (5)	COP **D-KUNDE** (3) (5)

Bemerkung: Die Variable FNAME liesse sich hier statt mit EQUATE auch mit dem Befehl DI '....' H W="FNAME" bilden.

b) Bilden einer Variablen DATUM mit dem System-Datum (vgl. auch unter dem Befehl WHEN):

```
EQ DATUM #DATE

Kontrolle:
DI SUB
   ergibt am 3.August 89:
   DATUM = 890803
```

Beispiele von EQUATE-Befehlen (Fortsetzung):

c) Löschen der Variablen DATUM: Das Wort "DATUM" wird nicht mehr ersetzt.

```
EQ DATUM
```

Kontrolle:

```
DI SUB
```

ergibt keine Ausgabe mehr für DATUM

d) Bilden einer zusammengesetzten Variable namens FILNAM aus "LOG" und dem Datum:

```
EQ DATUM  #DATE
EQ FILNAM LOG@DATUM
```

Kontrolle:

```
DI SUB
```

ergibt am 3.August 89:
DATUM = 890803
FILNAM = LOG890803

e) Bilden einer Variablen TEXT1 mit Leerstellen und Spezialzeichen im Inhalt. Verwendung für variable Meldungen.

```
EQ TEXT1 'WIEVIELE % PROVISION? '
```

f) Bilden und ausgeben einer Variablen AUSG, die aus 2 Text-Variablen und einem variablen Dateinamen (DNAME) zusammmengesetzt ist:

```
EQ TE1 'DATEI: '
EQ TE2 ' IST NICHT VORHANDEN!'

EQ AUSG TE1@DNAME@TE2

DI AUSG H          ; Ausgabe des Inhaltes von AUSG
```

(vgl. auch Controlstring-Befehl DISPLAY!)

g) Ermitteln der eigenen Bildschirm-Nummer (unter Verwendung der System-Variablen #PID):

```
EQ NUMMER #VALUE(#PID) / 100
      ; ergibt im Prozess 00302:  00302/100 = 3
```

(vgl. auch Controlstring-Befehl WHEN!)

h) EQUATE-Befehl mit Formel: Alle Generationen von 0 bis 10 der Datei S-WORK auf Disc 1 und 2 sind zu löschen. (vgl. auch Befehl WHEN)

```
EQ X 0
P1:
DEL S-WORK/X (1)
DEL S-WORK/X (2)
EQ X X + 1
WH X < 11 GO TO P1
```

i) Die Grösse einer Datei wird im Control-String berechnet. Die Anzahl Records als Berechnungs-Grundlage werden in einem DISPLAY-Befehl eingegeben:

```
DI 'ANZAHL DATEN-SAETZE: ' H W="REC"
EQ SECT REC * 64 / 512
AS D D-DEST (3) NE SECT
EX P-3 (2)
```

Wenn als "REC" 8000 eingegeben wird, läuft ab:

```
DI 'ANZAHL DATEN-SAETZE: ' H W="REC"
                              ;Eingabe = 8000
EQ SECT 8000 * 64 / 512
   Meldung: ... SECT = 1000
AS D D-DEST (3) NE 1000
EX P-3 (2)
```

Weitere Beispiele finden Sie unter den Befehlen WHEN und DISPLAY !

- Notizen -

EXECUTE (CONTROLSTRING)

Funktion:

Starten und Ausführen eines Controlstrings.

Ein analoges Format und gleiche Funktion als "Remote EXECUTE" hat der Befehl NETEXECUTE (Kapitel 4/NET).

Format:

```
EXECUTE physname / (n,DI) \
                 \ (n,MT) /
    [,NO]  [,LINK]  [substitutionen]  [START=zeile]
```

Parameter:

physname
: Controlstring-Name.

NO
: Durchlaufene Control-String-Befehle nicht am Bildschirm ausgeben, fakultativ.

  Vgl. auch Abschnitt "Anzeige der durchlaufenen Controlstring-Befehle".

LINK
: Nach Beendigung des aufgerufenen Control-String wieder in den aufrufenden Control-String zurückkehren und die Befehle nach diesem EX ausführen.
  Ohne LINK erfolgt keine Rückkehr in diesen Controlstring.

  Vgl. auch Abschnitt "Aufruf weiterer Controlstrings".

substitutionen
: Namen und Werte, welche alle Vorkommen der angegebenen Namen im Control-String ersetzen sollen.
  Die Substitutions-Angaben müssen zwischen Ä und Ü (bzw. zwischen eckigen Klammern) stehen.

  Vgl. auch Abschnitt "Substitutions-Parameter".

START=zeile
Der aufgerufene Controlstring wird erst ab der Zeile mit der Nummer "zeile" gestartet. Die erste Zeile ist Nummer 1, die zweite Nummer 2 usw.

Bemerkungen:

Anzeige der durchlaufenen Controlstring-Befehle während des Ablaufes

Ohne besondere Angaben werden alle ausgeführten Befehle eines Controlstrings am Bildschirm angezeigt.

Mit dem Parameter NO im startenden EXECUTE-Befehl kann diese Anzeige unterdrückt werden: Die Befehle werden nur ausgeführt; weder ausgeführte noch übersprungene Befehle werden angezeigt.

Mit dem Befehl LIST lässt sich im Controlstring selbst festlegen, ob keine Befehle, nur die ausgeführten oder auch die durch logische Entscheide übersprungenen Befehle anzuzeigen sind.
(vgl. Controlstring-Befehl LIST)

Aufruf weiterer Controlstrings

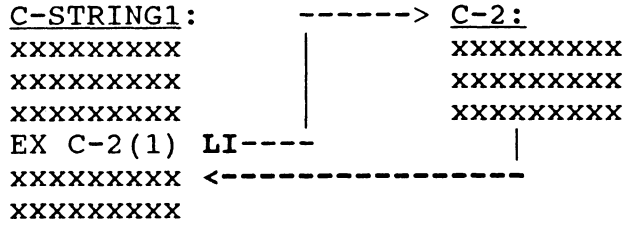
In einem Controlstring lassen sich mit EXECUTE beliebige weitere Controlstrings aufrufen.

Durch EX verkettete Cntrol-Strings bilden zusammen denselben Prozess. JCL-Code und Switch-Zustände werden von String zu String weitergegeben.

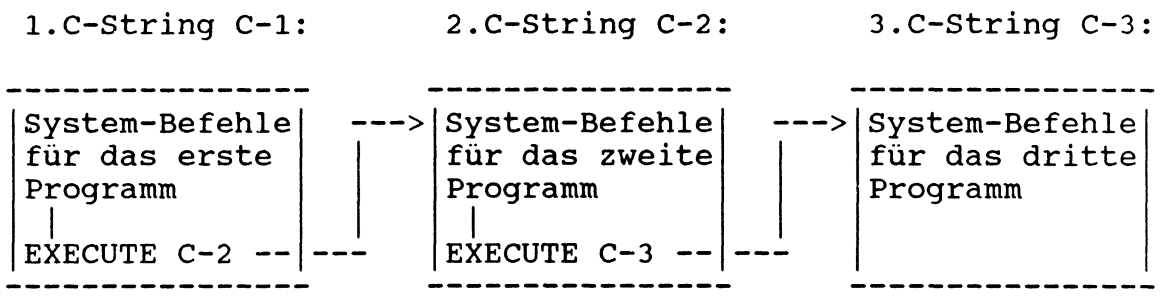
Ohne besondere Angabe erfolgt nach der Ausführung eines weiteren aufgerufenen Controlstrings keine Rückkehr in den Aufruf-String.

Mit dem LINK-Parameter im aufrufenden EXECUTE-Befehl wird bestimmt, dass die Steuerung am Ende des aufgerufenen Controlstrings wieder in den Aufruf-Controlstring zurückkehren soll.

Schema:



Mehrere Einstiege in einen Controlstring:
Um an mehreren Stellen in einen Controlstring einsteigen zu können, lässt sich ein langer Controlstring wie folgt aufteilen und mit EXECUTE-Befehlen am Ende jedes Teil-Controlstring verknüpfen:



EXECUTE C-1 führt den ganzen Controlstring (C-1 bis C-3) aus, EXECUTE C-2 den Controlstring ab dem zweiten Programm usw.

Uebergabe von Variablen an den aufgerufenen Controlstring:

Ohne besondere Angaben werden Variablen nicht in den aufgerufenen Controlstring übergeben.

Variablen-Uebergabe nur durch zusätzliche Substitutions-Angaben (vgl. Abschnitt "Substitutions-Angaben").

Variablen werden nicht an den Dialog nach der Controlstring-Ausführung zurückgegeben.

Bei Prozess-Ende werden alle Variablen gelöscht.

Beispiele (ohne Substitution):

a) Starten des Controlstrings C-A von Disc-Einheit 1:

```
EX C-A (1)
```

b) Starten des Controlstrings C-FIBU von Disc-Einheit 0. Durchlaufene System-Befehle nicht anzeigen.

```
EX C-FIBU (0) NO
```

c) Starten des Controlstrings C-T1 (4). Falls in diesem Control-String weitere Strings mit EX aufgerufen werden, soll bei deren Ende die Verarbeitung jeweils wieder in den Controlstring C-T1 zurückkehren:

```
EX C-T1 (4) LI
```

d) Starten des Controlstrings C-D (5) ab Zeile 6. Die ersten 5 Zeilen werden übersprungen.

```
EX C-D (5) ST=6
```

SUBSTITUTIONS-PARAMETER

In den System-Befehlen EX und SUB können Angaben in den aufgerufenen Control-Strings durch Substitutions-Angaben übersteuert werden:

Format:

/**EX**ECUTE\ physname (n) **Ä** /**alt = neu** \ [**alt = neu**] ... **Ü**
SUBMIT / \ **#ALL** /

Die Substitutions-Parameter müssen zwischen Ä und Ü bzw. in eckigen Klammern stehen (je nach Tastatur des Bidschirmes).

Funktion:

alt = neu

Beim Ablauf des Control-String werden alle Wörter im Controlstring, die gleich einer "alt"-Angabe sind, durch die zugeordnete "neu"-Zeichenfolge ersetzt.

Zum Beispiel:

EX C-1 (2) **Ä X=4 Ü**

Inhalt von C-1:	**läuft ab als:**
AS FA D1 (**X**)	AS FA DI (**4**)
EX P-1 (4)	EX P-1 (4)
DEL D1 (**X**)	DEL D1 (**4**)

#ALL

Durch Substitutions-Angaben werden auch Variablen in einen Controlstring übergeben.
Dabei übergibt die Angabe **#ALL** alle Variablen der laufenden Befehlsfolge in den aufgerufenen Controlstring.
(Vgl. auch den Abschnitt "Uebergabe von Variablen" einige Seiten weiter hinten.)

Regeln für alt- und neu-Zeichenfolgen:

Form	**zum Beispiel:**
Namen, bis 10-stellig entsprechend den Regeln für lognamen	X, FILE5, "A+B"
physnamen von Dateien, auch mit Generation	ASTAMM/1, BEW/
Gerät/Einheit-Angaben (Klammer-Inhalt)	(3), (LP), (1 LP)
physnamen mit Gerät/Einheit (ohne Leerstelle!)	WORK(3), FILE4/(7)
ganze Zahl, dezimal	3, 1500
ganze Zahl, hex	&3, &0FA1
@-Zahl Ersetzt alle Zahlen mit einem @-Zeichen davor. Dieselbe Zahl ohne @ wird nicht ersetzt.	@1, @20'
Zeichenfolge in Hochkommas (nur als "neu"-Parameter)	'ABLAUF MODIFIZIERT'

- Eine alt- oder neu-Zeichenfolge darf höchstens 30-stellig sein, mit folgenden Ausnahmen:
  - Namen höchstens 10-stellig
  - Zahlen höchstens = 2 147 483 647
  - Führende Nullen in Zahlen oder Leerstellen in einer Zeichenfolge zählen nicht (ausser in Zeichenfolgen zwischen Hochkommas).
  - Zeichenfolgen in Hochkommas dürfen über 30-stellig sein.
- Nach dem @-Zeichen darf keine Leerstelle folgen.

Bemerkungen:

Ohne besondere Angaben werden nur ganze Wörter ersetzt, keine Wort-Teile.
Zum Ersetzen von Wort-Teilen sind Wörter aus mehreren Variablen zusammenzusetzen, getrennt durch das Zeichen @.

Beim Ersetzen von Klammer-Ausdrücken (Geräte-Angaben) gelten die Angaben (5 DI), (5,DI) und (DI 5) als drei verschiedene Angaben, da nur die Zeichenfolge, nicht die Bedeutung untersucht wird.

Nicht ersetzt werden:
- Befehlscodes (z.B: AS, SET, EX usw.)
- Daten-Eingaben in Controlstrings, die mit SUBMIT ausgeführt werden.
- Sort-Parameter.

Eine entsprechende Technik mit Hilfe des Editors finden Sie am Ende des Kapitels "Submit-Batch-Verarbeitungen"

Beispiele von Substitutionen:

a) Uebersteuern eines Datei-Namens mit Disc-Einheit:

EX C-1 (1) **Ä F=D-JAN(4) Ü**

Inhalt von C-1:	es läuft ab als:
AS F1 **F**	AS F1 **D-JAN(4)**
EX P-1 (2)	EX P-1 (2)

Alternative:
Statt mit einer Substitution im aufrufenden EX-Befehl, könnte die Variable im Controlstring mit einem DISPLAY-Befehl verlangt werden:
DI ' FILENAME? ' H **W="F"**

b) Uebersteuern einer Disc-Einheit und eines Druckers:

```
EX C-2 (2) Ä (3)=(5) (1,LP)=(LP) Ü
 Keine Leerstellen in den "alt" oder "neu"-
 Zeichenfolgen!
```

Inhalt von C-2:	es läuft ab:
AS F1 FILE1 **(3)**	AS F1 FILE1 **(5)**
AS LO **(1,LP)**	AS LO **(LP)**
EX P-3 **(3)**	EX P-3 **(5)**

c) Uebersteuern einer Geräte-Angabe und einer Meldung:

```
EX C-3 (1)  Ä ()=(MT)  MELD='DATEN AB BAND' Ü
```

Inhalt von C-3:	es läuft ab:
AS F1 ZAHLUNG **()**	AS F1 ZAHLUNG **(MT)**
AS F2 ZAHL-D (3) NE 1000	AS F2 ZAHL-D (3) NE 1000
DI **MELD** H	DI **'DATEN AB BAND'** H
EX P-MELD3 (5)	EX P-MELD3 (5)

d) Uebersteuern einer Von-Bis-Angabe in einem DUMP-Befehl:

```
EX C-4 (1)  Ä V=10 B=20 Ü
```

Inhalt von C-4:	es läuft ab:
DI VFILE (3)	DI VFILE (3)
DU VFILE (3) **V B** (LP)	DU VFILE (3) **10 20** (LP)

Alternative:
Statt mit einer Substitution im aufrufenden EX-Befehl, könnten die Variablen im Controlstring mit je einem DISPLAY-Befehl verlangt werden:

```
 DI ' DUMP VON ? ' H W="V"
 DI ' DUMP BIS ? ' H W="B"
```

e) Unbeabsichtigtes Uebersteuern von Angaben, da unterschiedliche Dinge im Control-String durch die gleiche Zahl dargestellt sind:

EX C-5 (3) **Ä 1=4 Ü**

Inhalt von C-5:	es läuft ab:
AS A INFILE (**1**)	AS A INFILE (**4**)
AS B OUTFILE (2) NE 100	AS B OUTFILE (2) NE 100
EX P-5 (4)	EX P-5 (4)
DU OUTFILE (2) **1** 8 (LP)	DU OUTFILE (2) **4** 8 (LP)

! sollte nicht ersetzt werden!

f) Um im Beispiel e) nur die "1" im ersten Befehl (AS A INFILE) zu ersetzen, kann eine **@-Zahl** verwendet werden:

EX C-5 (3) **Ä @1=4 Ü**

Inhalt von C-5:	es läuft ab:
AS A INFILE (**@1**)	AS A INFILE (**4**)
AS B OUTFILE (2) NE 100	AS B OUTFILE (2) NE 100
EX P-5 (4)	EX P-5 (4)
DU OUTFILE (2) 1 8 (LP)	DU OUTFILE (2) 1 8 (LP)

Bemerkung: Wird dieser Control-String ohne Substitutions-Parameter aufgerufen, läuft er wie folgt ab:
@1 wird als 1 interpretiert.

AS A INFILE (1)
AS B OUTFILE (2) NE 100
EX P-5 (4)
DU OUTFILE (2) 1 8 (LP)

UEBERGABE VON VARIABLEN AN DEN AUFGERUFENEN CONTROLSTRING:

Ohne besondere Angaben werden Variablen nicht in den aufgerufenen Controlstring übergeben.

Variablen-Uebergabe nur durch zusätzliche Substitutions-Angaben:

```
EX c-string(n) Ä #ALL Ü
          ; übergibt alle Variablen an den c-string

EX c-string(n) Ä V1=V1 Ü
          ; übergibt nur die Variable V1

EX c-string(n) Ä V1=V1 V2=V2  Ü
          ; übergibt nur die Variablen V1 und V2
            an den c-string.

EX c-string(n) Ä V1=X1 Ü
          ; im aufgerufenen c-string erhält die
          Variable V1 den Wert der Variablen X1 des
          aufrufenden Controlstrings.
```

Variablen werden nicht an den Dialog nach der Controlstring-Ausführung zurückgegeben.

Bei Prozess-Ende werden alle Variablen gelöscht.

Beispiele für die Variablen-Uebergabe:

a) Die Variable X wird in den Controlstring C-3 übergeben:

Inhalt von C-2:	es läuft ab als:
DI 'Disc ? ' H W="X"	; Eingabe: 5
AS FA F1 (X)	AS FA F1 (5)
EX P-2 (1)	EX P-2 (1)
EX C-3 (2) **Ä X=X Ü**	
Inhalt von C-3:	**übernimmt die Variable X**
DEL WORK1 (**X**)	DEL WORK1 (**5**)
AS W WORK1 (**X**) NE 500	AS W WORK1 (**5**) NE 500
AS FA F1 (**X**)	AS FA F1 (**5**)
EX P-3 (1)	EX P-3 (1)

b) Alle Variablen des Controlstrings C-2 werden in den Controlstring C-3 übergeben:

Inhalt von C-2:	es läuft ab als:
DI 'File ? ' H W="**F**"	; Eingabe: D-ART
DI 'Disc ? ' H W="**X**"	; Eingabe: 5
AS FA F (X)	AS FA D-ART (5)
EX P-2 (1)	EX P-2 (1)
EX C-3 (2) **Ä #ALL Ü**	
Inhalt von C-3:	übernimmt die Variablen F und X:
DEL WORK1 (**X**)	DEL WORK1 (**5**)
AS W WORK1 (**X**) NE 500	AS W WORK1 (**5**) NE 500
AS FA **F** (**X**)	AS FA **D-ART** (**5**)
EX P-3 (1)	EX P-3 (1)

- Notizen -

EXIT (CONTROLSTRING)

Funktion:

Logisches Ende eines Control-Strings.
Dieser Befehl darf nur in Control-Strings verwendet werden.

Format:

EXIT [E]

Varianten:

ohne E — Der Control-String wird hier normal beendet mit der Meldung

```
CONTROLSTRING ..... DONE
```

Falls der laufende String von einem andern mit EX cstring (n) LINK aufgerufen wurde, kehrt die Steuerung in jenen String zurück.

mit E — Der Control-String wird hier abgebrochen mit der Meldung

```
CONTROLSTRING ..... ABORTED AT LINE n.
```

Bemerkungen:

Der EXIT-Befehl ist nicht erforderlich, wenn das physische auch das logische Ende des Control-Strings ist.

Im gleichen Control-String können keiner, einer oder mehrere EXIT-Befehle stehen.

Vorsicht im Menuprozessor bis Menu-Release 4.40.55:
(gilt für Menu-Releases ab 6.0 nicht mehr)

In einem Menu-Job bewirkt ein EXIT-Befehl den Austritt aus dem Menu-Prozessor!

Statt eines EXIT-Befehls ist ein Sprung auf einen Label am Job-Ende zu empfehlen:

statt:	--->	besser:
AS		AS .....
EX		EX .....
WH #JCL = 7 **EXIT**		WH #JCL = 7 **GO TO ENDE**
AS		AS .....
AS		AS .....
EX		EX .....
		ENDE:

GO (CONTROLSTRING)

Funktion:

Unbedingter Sprung in einem Controlstring.

Der GO-Befehl ist nur in Controlstrings erlaubt.

Format:

GO [TO] **label**

Parameter:

label — Label-Name eines System-Befehls im gleichen Controlstring. Der Controlstring wird bei jenem Befehl fortgesetzt.
Der Label-Name kann sich irgendwo im Controlstring befinden.
Ein Label kann auch, ohne dazugehörigen System-Befehl, am Ende eines Controlstring stehen.

Ein Label kann 1- bis 10-stellig sein. Er muss als erstes Zeichen einen Buchstaben enthalten. Nachher können Buchstaben, Ziffer und Bindestriche stehen. Er wird mit einem Doppelpunkt abgeschlossen.
Er kann auf einer eigenen Zeile oder vor einem System-Befehl stehen.

Beispiele:

a) Ueberspringen eines Controlstring-Teils:

```
IF 3 GO P3
AS .......
EX P-1 ...
GO TO ENDE
P3:                 ; Nach EX P-1 ...
AS .......          ; wird dieser
AS .......          ; Teil
EX P-2 ...          ; übersprungen
ENDE:
```

b) Variabler Sprung (entsprechend dem COBOL-Befehl **Perform**).
Tritt einer von 2 Fehlern auf, soll dieselbe Meldungs-Routine (MELDUNG) durchlaufen werden. Die Variable BACK wird als variable Label-Angabe verwendet:

```
EINGABE1:
EQ BACK EINGABE1
DI '1. DATEI ? ' H W=F1
WH .NOT. #FILE(F1(4))  GO MELDUNG
.......
.......
EINGABE2:
EQ BACK EINGABE2
DI '2. DATEI ? ' H W=F2
WH .NOT. #FILE(F2(6))  GO MELDUNG
.......
.......
; Gemeinsame Fehler-Routine:
MELDUNG:
DI 'DATEI NICHT VORHANDEN! ' H W
GO BACK                ; <--- modifizierter GO-Befehl:
                         Je nach Inhalt der Variablen
                         BACK: GO EINGABE1 oder
                               GO EINGABE2
```

Weitere Beispiele finden Sie unter den Controlstring-Befehlen IF und WHEN.

IF (CONTROLSTRING)

Funktion:

Abfragen des JCL-Codes und entsprechende Steuerung der Befehlsablauffolge in Controlstrings.

Der IF-Befehl ist nur in Controlstrings erlaubt.

Format:

```
IF [ /  [NOT] <  \ ]   w [,w] ...   [GO TO] label
   |    [NOT] =    |
   |      <>       |
   \    [NOT] >  / 
```

Parameter:

Bedingungen:

<	wenn der JCL-Code kleiner als w
>	wenn der JCL-Code grösser als w
<> oder NOT =	wenn der JCL-Code ungleich w
NOT <	wenn der JCL-Code nicht kleiner als w
NOT >	wenn der JCL-Code nicht grösser als w
leer oder =	gleich w
w	Vergleichswert für den JCL-Code. Werden mehrere Werte (mit oder ohne Komma dazwischen) angegeben, gilt die Oder-Verknüpfung.

label — Label-Name eines System-Befehls im gleichen Controlstring. Wenn die IF-Bedingung erfüllt ist, wird der Controlstring bei jenem Befehl fortgesetzt.
Der Label-Name kann sich irgendwo im Controlstring befinden.
Ein Label kann auch, ohne dazugehörigen System-Befehl, am Ende eines Controlstring stehen.

Ein Label kann 1- bis 10-stellig sein. Er muss als erstes Zeichen einen Buchstaben enthalten. Nachher können Buchstaben, Ziffern und Bindestriche stehen. Er wird mit einem Doppelpunkt abgeschlossen.
Er kann auf einer eigenen Zeile oder vor einem System-Befehl stehen.

Bemerkungen:

Das Betriebssystem sucht den Controlstring zuerst vorwärts bis zu seinem Ende ab. Anschliessend geht die Suche am Controlstring-Anfang weiter.

Anzeige übersprungener Befehle:
Wenn im Controlstring der Befehl LIST ALL aktiv ist, werden die übersprungenen Befehle auf dem Bildschirm mit einem Minuszeichen davor dargestellt.

Ein unbedingter Sprung (GO TO ohne Bedingung) kann mit GO programmiert werden:

```
GO TO label
```

(vgl. Controlstring-Befehl GO)

Hinweis:

Alle Varianten des IF-Befehl lassen sich auch mit einem WHEN-Befehl darstellen.
(vgl. Controlstring-Befehl WHEN)

Beispiele:

a) Falls der JCL-Code einen andern Wert als 0 aufweist, wird der SET SW-Befehl übersprungen.

```
IF <>  0  P1
SET SW 1 ON
P1:
AS .....
EX P-B(1)
```

Der obige IF-Befehl lässt sich auch darstellen als:

```
IF  <>    0   GO TO P1    oder
IF  NOT = 0   P1          oder
WH #JCL /= 1  GO TO P1    (vgl. Befehl WHEN)
```

b)
Falls der JCL-Code einen Wert kleiner als 4 enthält, wird der Controlstring beendet.

```
AS .........
EX P1
IF < 4  ENDE
AS .........
EX P2
ENDE:
```

Der obige IF-Befehl lässt sich auch darstellen als:

```
IF  < 4  GO TO ENDE      oder
WH #JCL  < 4 GO TO ENDE  (vgl. Befehl WHEN)
```

c) Falls zum Zeitpunkt des IF-Befehls der JCL-Code den Wert 1, 2 oder 4 enthält, erfolgt die Fortsetzung des Controlstring beim Befehl mit dem Label P6.

Enthält der JCL-Code einen andern Wert, werden die Befehle AS F0 ... und EX P-25 ausgeführt, anschliessend AS F1 ... und EX P-30.

```
AS ........
EX .........
IF 1 2 4  P6
AS F0 ......
EX P-25 (2)
P6:
AS F1 ...
EX P-30(2)
```

Der obige IF-Befehl lässt sich auch darstellen als:

```
IF  1 2 4  GO TO P6      oder
WH #JCL = 1 .OR. #JCL = 2 .OR. #JCL = 4 GO TO P6
  (vgl. Befehl WHEN)
```

d) Falls der JCL-Code = 1 ist: Das Programm P-100(2) , sonst das Programm P-200(2) ausführen. Darauf in jedem Fall das Programm P-ENDE ausführen.

```
IF 1  P100
AS ...
EX P-200(2)
GO P300            ;unbedingter Sprung
P100:
AS ...
EX P-100(2)
P300:
EX P-ENDE(3)
```

Der obige IF-Befehl lässt sich auch darstellen als:

```
IF  1  GO TO P100        oder
WH #JCL = 1 GO TO P100   (vgl. Befehl WHEN)
```

LIST (CONTROLSTRING)

Funktion:

Ein-/ausschalten der Ausgabe durchlaufener Controlstring-Befehle. Der Befehl gilt für den laufenden Controlstring und alle daraus aufgerufenen Controlstrings.

Format:

```
LIST  / ON  \
      | OFF |
      \ ALL /
```

Parameter:

ON — Alle durchlaufenen Controlstring-Befehle und Kommentare werden ausgegeben, **ohne** die mit IF oder WH übersprungenen.
Entspricht dem EX-Befehl ohne NO-Parameter.

OFF — Die durchlaufenen und übersprungenen Controlstring-Befehle und Kommentare werden **nicht** ausgegeben.
Entspricht dem NO-Parameter im EX-Befehl.

ALL — Alle durchlaufenen Controlstring-Befehle und Kommentare werden ausgegeben, **auch** die mit GO, IF oder WH übersprungenen.

Bemerkungen:

Der List-Befehl im einem Controlstring übersteuert den NO-Parameter im startenden EXECUTE-Befehl.

Er erlaubt den mehrmaligen Wechsel des Ausgabemodus innerhalb eines Controlstrings.

Beispiel:

```
LIST ON
;       Nur die ausgeführten Befehle und Kommentare
;       werden angezeigt
WH #WKDAY = MONDAY GO TO P2
; AM MONTAG NICHT AUSFUEHEREN:
AS ......
EX P-100 ...
P2:
; AN ALLEN TAGEN AUSFUEHREN:
LIST ALL
;       alle Befehle und Kommentare
;       werden angezeigt, die übersprungenen
;       mit einem Minus-Zeichen davor
IF > 0 GO TO P3
AS ......
EX P-200 ...
P3:
LIST OFF
;       Keine Befehle und Kommentare
;       werden angezeigt
IF = 7 GO TO P9
AS ......
EX ......
P9:
LIST ALL
;       Wieder alle Befehle und Kommentare
;       anzeigen
; ENDE DES BEISPIELS
```

NOABORT (CONTROLSTRING)

Funktion:

Festlegen, ob ein Controlstring-Abbruch erfolgt, wenn im Controlstring ein Fehler auftritt.
Der NOABORT-Befehl gilt nur für den Controlstring, in dem er steht. Er wird nicht in Strings übernommen, die von diesem aus aufgerufen werden.

Format:

NOABORT [/ ON \ ].
\ **OFF** /

Parameter:

ON Bei einem Fehler im Controlstring erfolgt kein Abbruch.
Die Fehlermeldung wird ausgegeben und der Controlstring läuft mit dem nächsten Befehl weiter.

OFF Bei einem Fehler im Controlstring wird dieser nach Ausgabe der entsprechenden Fehlermeldung abgebrochen.

NOABORT **ohne** Parameter
ist gleichbedeutend wie NOABORT ON.

Enthält ein Controlstring **keinen NOABORT-Befehl**, gilt NOABORT OFF.

Bemerkungen:

Nach jedem Befehl des Controlstrings kann mit der Funktion **#ERROR** festgestellt werden, ob der Befehl fehlerfrei abgelaufen ist.

Wichtig:
Der Test auf #ERROR muss unmittelbar nach dem Befehl erfolgen. Dazwischen ist auch keine Kommentarzeile erlaubt!:

WH #ERROR
prüft, ab der letzte Befehl fehlerfrei war
oder:
EQ variable #ERROR
speichert #TRUE oder #FALSE in einer Variablen, die auch später im Controlstring getestet werden kann.
Bei Fehler wird #TRUE gespeichert.
Ein fehlerfreier Befehl ergibt #FALSE.

Wenn NOABORT ON gilt, meldet der Befehl EXIT E "Controlstring DONE", nicht "Controlstring Aborted"-

Beispiele:

In einem Controlstring mit direkter Drucker-Zuteilung ist zu prüfen, ob der Drucker frei ist:

a) Bei besetztem Drucker automatisch ein manuelles Spoolfile zuteilen (vereinfachte Variante):

```
NOABORT          ; oder NOABORT ON
AS LO (3 DLP)
WH #ERROR   AS LO SP-3DLP(0) NE 100 SP AP
AS KUND D-KUNDE (3)
EX P-DRU (2)
```

Der Befehl WH #ERROR muss unmittelbar nach dem zu testenden Befehl AS LO stehen!

Dieses Beispiel testet jedoch nicht, ob das zu erstellende Spoolfile schon vorhanden ist und der Benutzer wird nicht informiert, ob ein Spoolfile erstellt wird.
Ein umfassenderes Beispiel finden Sie als Beispiel b)

In einem Controlstring mit direkter Drucker-Zuteilung ist zu prüfen, ob der Drucker frei ist:

b) Bei besetztem Drucker ein mauelles Spoolfile eröffnen und melden, dass ein Spoolfile gebildet wurde (anspruchsvollere Variante):

```
NOABORT              ; oder  NOABORT ON
AS LO (3 DLP)
EQ BESETZT #ERROR ; Variable BESETZT bilden
NOABORT OFF          ; bei andern Fehlern abbrechen
WH BESETZT = #FALSE GO TO WEITER
;
                     ; Spoolfile eröffnen statt 3,DLP:
DEL SP-3DLP (0)
AS LO SP-3DLP (0) NE 100 SP AP
                     ; Information des Benutzers:
DI 'LISTE IN SP-3DLP(0)' H W
;
WEITER:
AS KUND D-KUNDE (3)
EX P-DRU (2)
```

Bemerkung:
Weil der Sachverhalt "Fehler im Befehl AS LO " auch später als unmittelbar nach dem AS-Befehl noch zu testen ist, müss eine Variable gebildet werden.

- Notizen -

SET JCL

Funktion:

Setzen des JCL-Codes.

Format:

SET JCL TO w

w : Der JCL-Code der Verarbeitung wird auf diesen Wert gesetzt (0 bis 7).

Bemerkungen:

Der JCL-Code kann mit den System-Befehlen IF und WHEN abgefragt werden.
Der JCL-Code bleibt für die Dauer des Prozesses (gleiche Prozess-Identifikation) erhalten, sofern er nicht durch Programme oder SET JCL-Befehle verändert wird.

Beispiel:

Den JCL-Code auf 2 setzen:

```
SE  JC  2
```

- Notizen -

WHEN (CONTROLSTRING)

Funktion:

Von einem arithmetischen oder boolschen Ausdruck abhängige Steuerung der Ablauffolge in Controlstrings.

Der Befehl ist nur in Controlstrings erlaubt.

Format:

WHEN ausdruck / GO TO label
\ System-Befehl/

ausdruck: Bedingung in Form einer "Formel" (siehe Abschnitt "Bedingungen im WHEN-Befehl")

label: Label-Name eines System-Befehls im gleichen Control-String. Wenn die IF-Bedingung erfüllt ist, wird der Control-String bei jenem Befehl fortgesetzt. Der Label-Name kann sich irgendwo im Control-String befinden. Ein Label kann auch, ohne dazugehörigen System-Befehl, am Ende eines Control-String stehen.

Ein Label kann 1- bis 10-stellig sein. Er muss als erstes Zeichen einen Buchstaben enthalten. Nachher können Buchstaben, Ziffer und Bindestriche stehen. Er wird mit einem Doppelpunkt abgeschlossen. Er kann auf einer eigenen Zeile oder vor einem System-Befehl stehen.

System-Befehl:
Ein einzelner System-Befehl, der nur ausgeführt wird, wenn der "ausdruck" wahr ist.

Bemerkungen:

Ist die Bedingung erfüllt, sucht das Betriebssystem den Control-String zuerst vorwärts bis zu seinem Ende ab. Anschliessend geht die Suche am Controlstring-Anfang weiter.

Anzeige übersprungener Befehle:
Wenn im Controlstring der Befehl LIST ALL aktiv ist, werden die übersprungenen Befehle auf dem Bildschirm mit einem Minuszeichen davor dargestellt.

Ein unbedingter Sprung (GO TO ohne Bedingung) kann mit GO programmiert werden:

GO TO label

(vgl. Controlstring-Befehl GO)

BEDINGUNGEN im WHEN-Befehl:

Operanden und Operatoren in "ausdruck":

Ein "ausdruck" kann sich zusammensetzen aus:

- Variablen-Namen (siehe unter dem Befehl EQUATE)
- Konstanten
- SCL-Funktionen (Ausdrücke mit #....)
- Arithmetischen, relationalen und boolschen Operationen

Variablen-Namen:

Ein Wort von 1 bis 10 Zeichen, beginnend mit einem Buchstaben. Die übrigen Zeichen können Buchstaben, Ziffern oder eingeschlossene Bindestriche sein.

Beachten Sie den Abschnitt "Variablen" am Anfang des Kapitels "Control-Strings".

Numerische Konstanten:

Ganze Zahlen von 0 bis 2 147 483 647.

Alphanumerische Konstanten:

Müssen zwischen Anführungszeichen gesetzt werden, z.B: "X"

Mit dem Parameter W="variable" des Befehls DISPLAY gebildete Variablen gelten als alphanumerisch!

SCL-Funktionen

Können auch mit EQUATE angesprochen werden!

#AS (logname) — Ist wahr, wenn dem eigenen Prozess eine Datei mit dem lognamen assigned ist.

Beispiel:
```
WH #AS(A) GO TO P1
```
Wenn im eigenen Prozess der logname A assigned ist.

#BATCH — Ist wahr, wenn der eigene Prozess als Submit-Batch-Prozess abläuft.

Beispiel:
```
WH #BATCH GO TO PBATCH
```

#COB(n) — Ist wahr, wenn der Switch-n ON ist.

Beispiel:
```
WH #COB(2) GO TO P0
```
Wenn Switch-2 ON ist.

#DATE — Ergibt das aktuelle Datum, 6-stellig, in der Form JJMMTT.

Beispiel:
```
WH #DATE < 900201 GO TO P3
```
Wenn heute noch nicht der 1.2. 1990 ist.

SCL-Funktionen (Fortsetzung)

#DAY — Ergibt den aktuellen Tag, 1- oder 2-stellig.

Beispiel:
```
WH #DAY = 15 GO TO P3
```
Wenn heute der 15. des Monats ist.

#ERROR — Testet, ob in der unmittelbar vorangehenden Zeile ein Fehler festgestellt wurde und ergibt #TRUE oder #FALSE. Gilt auch für Fehler, die zu keinem Abbruch führen würden.
---> vgl. Befehl NOABORT!

#FILE(physname(n)) — Ist wahr, wenn eine Datei mit dem Namen "physname" auf Disc n vorhanden ist.

Beispiel:
```
WH #FILE(D-TEM(4)) GO TO P4
```
Wenn das File D-TEM auf Disc 4 existiert.

#JCL — Ergibt den Wert des JCL-Code als ganze Zahl.

Beispiel:
```
WH #JCL = 1 GO TO P4
```
Wenn der Wert des JCL-Code gleich 1 ist.

#MONTH — Ergibt den aktuellen Monat, 1- oder 2-stellig.

Beispiel:
```
WH #MONTH = 12 GO TO P4
```
Wenn heute Dezember ist.

#NUM(x) — Ist wahr, wenn die Variable in der Klammer (x) einen numerischen Inhalt hat.

Beispiel:
```
DI 'DISC-UNIT? ' H W="DISK"
WH #NUM(DISK) GO TO P5
```
Wenn die Variable DISK einen numerischen Inhalt aufweist.

SCL-Funktionen (Fortsetzung)

#PID — Ergibt die Prozess-Identifikation des eigenen Prozesses in der Form xxxyy (ohne Punkt).

Beispiel:

```
WH #VALUE(#PID) < 00600 .OR. !
   #VALUE(#PID) > 00899 GO TO P9
```

Wenn der Control-String nicht am Schirm 6 bis 8 läuft (Prozess < 006.00 oder > 008.99).

#SIZE(physname(n)) — Ergibt die Sektoren-Anzahl, die dem File "physname" auf Disc n zugeteilt sind.

Beispiel:

```
WH #SIZE(D-SAMMEL(2)) > 1000 GO TO P5
```

Wenn das File D-SAMMEL auf Disc 2 über 1000 Sektoren gross ist.

#TIME — Ergibt die Uhrzeit, 4-stellig, in der Form STMI.

Beispiel:

```
WH #TIME < 1200 GO TO P6
```

Wenn es noch nicht 12 Uhr mittags ist.

#USER — Ergibt den USER aus der Anmeldung, wenn $ACCESS aktiviert ist.
Ist $ACCESS nicht aktiviert, heisst der User TERMnnn, wobei nnn die Bildschirmnummer darstellt.

Beispiele:

```
WH #USER = TERM010 GO TO P10
```

Wenn der laufende Prozess am Bildschirm 10 abläuft (System ohne $ACCESS).

```
WH #USER = "HMS" GO TO P10
```

Wenn der angemeldete Benützer "HMS" heisst. (System mit $ACCESS).

SCL-Funktionen (Fortsetzung)

#VALUE(x) Ergibt den aktuellen Wert der Variablen x.

Beispiel:

```
DI 'OK (J/N)? ' H W="ANTW"
WH #VALUE(ANTW) = "J" GO TO P5
```

Wenn die Variable ANTW den Inhalt "J" aufweist.

Test auf die Eingabe "<RET>-Taste":

```
WH #VALUE(ANTW) = '' GO TO P5
```

(Zwei Hochkommas)

#WKDAY Ergibt den aktuellen Wochentag in der Form: MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY oder SUNDAY.

Beispiel:

```
WH #WKDAY = MONDAY GO TO P7
```

Wenn es heute Montag ist.

#YEAR Ergibt das aktuelle Jahr, 4-stellig.

Beispiel:

```
WH #YEAR  > 1989 GO TO P7
```

Wenn das Jahr 1989 schon vorbei ist.

SCL-Funktionen lassen sich mit dem Befehl EQATE auch in Variablen speichern und als Variable weiterverarbeiten und kombinieren.

Beispiel:

```
EQ TAG #DAY       ; ergibt am 12.8. den Wert 12
EQ MON #MONTH     ; ergibt am 12.8. den Wert 8
EQ FIX LOG        ; ergibt den Fix-Wert "LOG"

EQ MOTA FIX@TAG@MON  ; ergibt Wert LOG812
```

Vgl. auch unter dem Befehl EQUATE.

Arithmetische Operationen:

Addition	:	+	Vor und nach jedem
Subtraktion	:	-	Operationszeichen
Multiplikation	:	*	muss mindestens eine
Division	:	/	Leerstelle stehen.
Exponent (hoch)	:	^	

Die arithmetische Funktionen arbeiten nur mit ganzen Zahlen.

+ und - können auch einem Namen oder Ausdruck in Klammern vorangestellt werden:
Ein vorangestelltes - bedeutet : multipliziert mit -1.

Bei Ketten mit mehr als 2 Operanden werden Klammern empfohlen!

Beispiele arithmetischer Ausdrücke:
(A, B, C und D seien Variablen, die mit EQUATE gebildet wurden)

```
A + 2

(B +C) * 3

A / 4 * -D

X * ( -(A + B) + #JCL )
```

Relationale Operationen:

gleich	=
grösser als	>
kleiner als	<
ungleich	/=
grösser/gleich	>=
kleiner/gleich	<=

Vor und nach jedem Operationszeichen muss mindestens eine Leerstelle stehen.

Boolsche Operationen:

boolsch gleich	.EQ.
und	.AND.
oder	.OR.
ungleich	.NE.
nicht	.NOT.

Enthält ein Ausdruck mindestens 1 boolsche Operation, gilt er als boolsch.

Das Ergebnis von boolschen Operationen ist immer:

#TRUE (wahr) oder
#FALSE (falsch).

Werden verschiedene Operatoren im gleichen Ausdruck verwendet, löst das Betriebssystem den Ausdruck von links nach rechts auf, sofern nicht die folgenden Prioritätsregeln eine andere Auflösung bewirken.

Bei Ketten mit mehr als 2 Operanden werden Klammern empfohlen!

1. Stufe: .NOT.und vorangesetztes + oder -

2. Stufe: ^ (hoch)

3. Stufe: =, /=, <, >, <=, >=, .EQ., .NE.

4. Stufe: *, / und .AND.

5. Stufe: +, - und .OR.

Beispiele von arithmetischen und boolschen Ausdrücken :

A + B * 3	ergibt:	A + (B * 3)
A .AND. B .EQ. 4		A .AND. (B .EQ. 4)
A .NOT..EQ. B .OR. C		(A .NOT. .EQ. B) .OR. C
B < C .AND. X = 4		(B < C) .AND. (X = 4)

Beispiele von WHEN-Befehlen:

a) Das Programm P-1 samt seinen Datei-Zuteilungen überspringen, wenn SWITCH-1 und 3 ON sind:

```
WH #COB(1) .AND. #COB(3) GO TO P2
AS DK ...
AS TXT ...
EX P-1 (2)
P2:
AS F1 ....
EX P-2 (2)
```

b) Das Programm P-3 verwendet eine bestehende Datei D-3 (4). Falls diese Datei noch nicht besteht, soll eine neue Datei eröffnet werden.
Logik mit einer "Entweder-oder-Verknüpfung".

```
WH #FILE(D-3(4)) GO TO ALT
;    neue Datei eröffnen:
AS F1 D-3 (4) NE 500
GO TO PROG
ALT:
;    bestehende Datei verwenden:
AS F1 D-3 (4)
;    Programm-Start:
PROG:
EX P-3 (1)
```

c) Alle Generationen von 0 bis 10 der Datei S-WORK auf Disc 1 und 2 sind zu löschen:
Bildung einer logischen Schleife.

```
EQ X 0
P1:
DEL S-WORK/X (1)
DEL S-WORK/X (2)
EQ X X+1
WH X < 11 GO TO P1
DI 'Löschen beendet ' H
```

d) Das Programm P-4 nur ausführen, wenn der JCL-Code den Wert 3 und die Variable V4 den Wert "JA" aufweist:

```
DI 'P-4 AUSFUEHREN, WENN JCL-CODE=3 ? ' H W="V4"
WH .NOT. (#JCL = 3 .AND. V4 = "JA") GO TO P4
AS ....
EX P-4 (2)
P4:
.....
```

Beachten Sie die Klammer um den ganzen mit .NOT. negierten Ausdruck.

e) Das Programm P-5 soll so oft wiederholt ablaufen, bis der JCL-Code = 0 ist, jedoch höchstens 3 x (Logische Schleife).

```
;  Zähler Z auf 1 setzen:
EQ Z 1
P4:
WH #JCL = 0 GO TO ENDE
EX P-5(1)
;  Zähler um 1 erhöhen und auf 3 testen:
EQ Z Z+1
WH Z <= 3 GO TO P4
ENDE:
```

f) Ein Programm nur am Samstag zwischen 8 und 16 Uhr ausführen (Dreifach-Bedingung):

```
WH #WKDAY = SATURDAY .AND. !
( #TIME > 0759 .AND. #TIME < 1601) !
 GO TO AUSF
DI ' JOB JETZT NICHT ERLAUBT ' H
GO P9
AUSF:
AS ......
EX P-....
P9:
```

Die Klammern sind erforderlich!

Die Zeichen ! an den Zeilenenden sind erforderlich, weil sich der WHEN-Befehl über mehrere Zeilen erstreckt.

Weitere Beispiele finden Sie unter dem Controlstring-Befehl EQUATE.

- Notizen -

22. MULTI-SECTION / MULTI-VOLUME -FILES

Dieses Kapitel umfasst alle System-Befehle für Dateien, welche aus mehreren Sections bestehen können.

22.1 Möglichkeiten

Datenfiles und manuelle Spoolfiles können aus maximal 50 Sections bestehen. Alle Sections einer Datei sind gleich gross und können sich auf derselben Disc-Einheit oder auf verschiedenen logischen Disc-Einheiten befinden.

Beim Lesen oder Kopieren verknüpft die Systemsoftware die Sections einer Datei automatisch. Die Sections einer Datei sind nicht einzeln ansprechbar.

Einschränkungen:

Disc-Format:
Multi-Section-Files sind nur auf Disceinheiten möglich, die als Format 10 initialisiert sind.
Alle Disceinheiten, wo sich Sections derselben Datei befinden, müssen im gleichen Sektor-Format initialisiert sein (alle Full- oder alle Short-Sector).

Datei-Typen:
Nur Indexdateien mit separatem Index ("New Style") sind in Sections aufteilbar; "Old Style"-Indexdateien mit integriertem Index müssen immer aus einer einzigen Section bestehen.

Object-Programme sind nicht in Sections aufteilbar.

$STREAM:
Multi-Volume-Dateien (Dateien, deren Sections auf mehrere Disc-Einheiten verteilt sind), lassen sich zur Zeit nicht mit $STREAM auf Streamer-Band kopieren.

Somit sind Multi-Volume-Files auf Systemen ohne Magnetband nicht verwendbar!

22.2 ERFORDERLICHE SYSTEM-BEFEHLE (UEBERSICHT)

Alle Sections auf derselben Disceinheit:

Erfordern bei der Zuteilung einen **ASSIGN**-Befehl mit dem **PLACE-Parameter SEC oder SAME.**

Im Disc-Directory sind die einzelnen Sections nur sichtbar, wenn ein **DI-Befehl mit dem Parameter E oder FU** ausgeführt wird.

Sections auf mehrere Disc-Einheiten verteilt (Multi-Volume-Dateien):

GROUP-Befehl:
Mit dem GROUP-Befehl sind zuerst **Gruppen von logischen Disc-Einheiten** zu bilden. Die Sections von Multi-Volume-Dateien können nur auf die Disc-Einheiten derselben Gruppe verteilt werden.
Dabei lassen sich auch Verkettungen aufbauen,
z.B Disc 1 --> Disc 3 --> Disc 4 usw.
(Kreislauf wiederum ab Disc 1...)

Eine bestimmte logische Disc-Einheit kann nur einer einzigen Gruppe angehören.

Kontrolle der GROUP-Defintionen:
Mit dem Befehl **DISPLAY GROUP.**

ASSIGN-Befehl:
Im ASSIGN-Befehl wird mit einem PLACE-Parameter angegeben, wo weitere neue Folge-Sections zu bilden sind (unabhängig von der Platzierung bestehender Sections):

- **SEC oder SAME:**
  Auf derselben logischen Einheit, wo sich die erste Section befindet.

- **SPREAD:**
  Alternierend auf allen Disc-Einheiten, die mit GROUP als Gruppe von Disc-Einheiten festgelegt wurden.

- **SPACE:**
  Auf derselben logischen Einheit, wo sich die erste Section befindet. Falls jedoch dort kein Platz mehr vorhanden ist, auf der nächsten Disc-Einheit gemäss GROUP-Definition.

Befehle, die eine Multi-Section-Datei ansprechen:

In allen Befehle, die eine Datei ansprechen (AS, DI CHA, DEL, MOV usw.) ist die Disc-Einheit anzugeben, auf der sich die erste Section der Datei befindet.

Im Disc-Directory sind die einzelnen Sections nur sichtbar, wenn ein DI-Befehl mit dem Parameter E oder FU ausgeführt wird.

Index-Dateien:

Multi-Section-Index-Dateien sind nur als "New Style"-Dateien mit separatem Index möglich. Die erste Index-Section muss dabei immer auf der Disceinheit der ersten Daten-Section Platz finden.

Bei der Definition einer Disc-Gruppe mit GROUP oder beim Ansprechen einer Multi-Volume-Datei müssen alle betreffenden Disc-Einheiten **mounted** sein.

Kopieren und Löschen ganzer Disc-Einheiten:

Vorsicht! Bei Kopieren oder Löschen ganzer Disc-Einheiten mit Multi-Section-Files ist Vorsicht geboten:
---> Vgl. Abschnitte DELETE und MOVE !

GROUP und DISPLAY GROUP

Funktion:

Festlegen von Gruppen logischer Disc-Einheiten für Folge-Sections von Dateien.

Folge-Sections einer Datei können nur auf Disceinheiten derselben Gruppe liegen.
Jede Gruppe besteht aus einer Primär- und 0 bis mehreren Sekundär-Einheiten.
Eine bestimmte logische Einheit darf nur in einer einzigen GROUP-Definition vorkommen.
Dabei lassen sich jedoch Verkettungen aufbauen,
z.B Disc 1 ---> Disc 3 ---> Disc 4
(Kreislauf wiederum ab Disc 1)

Formate:

a) Bilden, ändern, löschen von Disc-Gruppen-Zuteilungen:

```
GROUP prim / CR \ sek [sek]...
           ! AD !
           \ DE /
```

b) Löschen einer ganzen Gruppe:

```
GROUP prim  DE ALL
```

c) Kontrolle der aktuellen Disc-Gruppen-Definitionen:

```
GROUP DI  / n [ n ]...\  [ TO ( / n \,/LP \ ) ]
          \ ALL       /          \ANY/ \DLP/
oder:

DISPLAY GROUP / n [ n ]...\  [ TO ausgabegerät ]
              \ ALL       /
```

Parameter:

prim	Primär-Disceinheit einer Gruppe, ohne Klammern.
sek	Sekundär-Disceinheit(en) einer Gruppe, ohne Klammern.
n	Eine Disceinheit, ohne Klammern.
ALL	Alle Disc-Einheiten, die einer Gruppe angehören.
CR	Mit dieser Variante wird eine neue Gruppe gebildet. Dies ist nur möglich, wenn die Primär-Disceinheit noch keiner GROUP-Definition angehört.
AD	Mit dieser Variante wird eine bestehende Gruppe um eine oder mehrere Sekundär-Disceinheiten erweitert.
DE	Mit dieser Variante werden eine oder mehrere Disc-Einheiten aus einer bestehenden Gruppe gelöscht. Dies ist nur möglich, wenn die zu löschenden Disc-Einheiten kein Folge-Sections von Dateien enthalten, deren Primär-Section auf einer andern Einheit liegt.
TO (/ n \,/LP \) \ANY/ \DLP/	Ausgabe auf einem Drucker. Ohne diesen Parameter erscheint die Ausgabe auf dem Bildschirm.
ausgabegerät	Wie in allen DISPLAY-Befehlen (vgl. Kapitel 4/DIS).

Bemerkungen:

Einer Gruppe können höchstens 16 logische Disc-Einheiten angehören.
Eine bestimmte logische Einheit darf nur in einer einzigen GROUP-Definition vorkommen.
Die Reihenfolge der Folge-Disceinheiten für alle Multi-Section-Dateien innerhalb einer GROUP ist stets die Reihenfolge, in der die Einheiten der Gruppe definiert wurden. Die Folge-Disceinheit der letzten Sekundär-Einheit ist wiederum die Primär-Einheit.
Kontrolle: Mit dem Befehl GR DI.

Auf jeder Disc-Einheit, die einer GROUP angehört, wird ein Systemfile ((GROUP)) gebildet, das die entsprechenden Definitionen enthält.
Dieses Systemfile veschwindet wieder, sobald die Disceinheit keiner Gruppe mehr angehört.
Der Befehl GR prim CR ... ist nur mögich, wenn die Disceinheit "prim" noch keiner GROUP angehört und somit noch kein File ((GROUP)) enthält.

Vorsicht mit Packnummern:
In der Gruppen-Definition werden neben der Einheits-Nummer auch die Packnummern der Discs abgespeichert.
Dieselbe Packnummer darf nicht mehrmals auf dem System mounted sein!

Löschen/Aendern von GROUP-Zuteilungen:

Disc-Einheiten, auf denen Folge-Sections bestehen, können nicht aus einer Gruppe gelöscht werden.

Die Löschung einer Einheit aus einer GROUP-Definition wird bestätigt mit der Meldung:
UNIT n DELETED
Wird eine Sekundär-Einheit aus einer Gruppe gelöscht und enthält die vorangehende Einheit der Gruppe aktive ASSIGN's mit SPACE oder SPREAD, jedoch ohne bestehende Folge-Sections auf der Sekundär-Einheit, erfolgt zusätzlich die Warnung:
WARNING: FILES ARE ASSIGNED

Die ganze Gruppen-Definition kann nur gelöscht werden, wenn keine Folge-Sections von Dateien auf Sekundär-Einheiten bestehen.
Die Löschung einer ganzen GROUP-Definition wird bestätigt mit der Meldung:
UNIT n GROUP DISSOLVED

Beispiele:

a) Neu bilden einer Gruppe aus den Disc-Einheiten 3 (Primär-Einheit) und 2. Beide Einheiten dürfen noch keiner Gruppe angehören:

```
GR 3 CR 2
```

b) Erweitern der Gruppe mit Primär-Einheit 3 um Disc-Einheit 4: Disceinheit 4 wird die letzte Sekundär-Einheit der Gruppe.

```
GR 3 AD 4
```

c) Entfernen der Einheit 4 aus dieser Gruppe. Einheit 4 darf in diesem Zeitpunkt keine Folge-Sections von Dateien auf andern Einheiten enthalten:

```
GR 3 DE 4
```

d) Löschen einer ganzen Gruppen-Definition, die Disc-Einheit 7 als Primär-Einheit hat. Zu diesem Zeitpunkt dürfen keine Multi-Volume-Files in dieser Gruppe bestehen:

```
GR 7 DE ALL
```

e) Ausgabe der Gruppen-Zuteilung, der Disc-Einheit 1 angehört, auf den Bildschirm:

```
GR DI 1          oder:     DI GR 1
```

f) Ausgabe sämtlicher Gruppen-Zuteilungen auf den Drucker 0:

```
GR DI ALL (LP)   oder:     DI GR ALL (LP)
```

Ausgabe von GR DI ... und DI GR ...

Am Beispiel GR DI ALL bzw. DI GR ALL für zwei Gruppen, die gebildet wurden mit:

```
GR 1 CR 2 3      und
GR 4 CR 0
```

```
-----------------------------------------
|                                       |
|    VSN     UNIT                       |
|    100001   1       PRIMARY           |
|    100003   2       100001    1       |
|    100003   3       100001    1       |
|                                       |
|    100004   4       PRIMARY           |
|    100000   0       100004    4       |
|                                       |
-----------------------------------------
```

Jede Zeile stellt eine Disc-Einheit dar, die zu einer GROUP gehört. Die Reihenfolge der Zeilen entspricht der Reihenfolge der Disc-Einheiten innerhalb der GROUP.

Links:
Packnr/Einheit der betreffenden logischen Platteneinheit.

Rechts:
Packnr/Einheit der Primär-Einheit der GROUP.
PRIMARY = Dies ist die Primär-Einheit selbst.

ASSIGN (Multi-Section-Dateien)

Dieses Kapitel enthält nur die Parameter, die für Disc-Multisection-Dateien verwendet werden.

In ASSIGN-Befehlen für Disc-Dateien kann jeweils einer dieser PLACE-Parameter irgendwo nach der Gerätezuteilung (Klammern) stehen.

Alle diese PLACE-Parameter gelten nur für neue Sections, die während der betreffenden Zuteilung der Datei gebildet werden. Auf schon bestehende Sections haben sie keinen Einfluss.
Wenn der Parameter SAME, SPACE oder SPREAD bei einer Zuteilung gegenüber früherer Zuteilungen verändert wird, gilt jeweils der zuletzt angegebene.

Im Disc-Directory wird der aktuelle Parameter für jede Datei abgespeichert im Feld PLACE. Dieses Feld ist nur sichtbar, wenn ein DI-Befehl mit dem Parameter E oder FU ausgeführt wird.

SEC

Neue Sections werden auf derselben logischen Einheit gebildet, wo sich die erste Section befindet.

Der SEC-Parameter lässt bei bestehenden Dateien früher festgelegte SAME, SPREAD oder SPACE-Parameter im Disc-Directory unverändert. (Unterschied zu SAME !)

Achtung: Bei shared Dateien bestimmt der erste Assign, ob SEC gilt oder nicht.
Empfehlung: Für solche Dateien in allen Assigns SEC angeben.

SAME

Neue Sections werden auf derselben logischen Einheit gebildet, wo sich die erste Section befindet.

Der SAME-Parameter übersteuert früher festgelegte SPREAD oder SPACE-Parameter.
In der Directory-Eintragung "PLACE" wird SAME eingesetzt. (Unterschied zu SEC !)

SPACE

Neue Sections werden auf derselben logischen Einheit gebildet, wo sich die erste Section befindet. Falls dort kein Platz mehr für eine neue Section vorhanden ist, wird die nächste Section auf der nächsten Disceinheit der Gruppe eröffnet.

Der SPACE-Parameter übersteuert früher festgelegte SAME- oder SPREAD-Parameter.
In der Directory-Eintragung "PLACE" wird SPACE eingesetzt.

SPREAD

Neue Sections werden alternierend auf allen Disc-Einheiten der Gruppe gebildet.
Beispiel:
Als Gruppe wurde definiert: GR 1 CR 3 4
Die erste Section wird auf Disc 1, die zweite auf Disc 3, die dritte auf Disc 4, die vierte wiederum auf Disc 1 erstellt usw.

Der SPREAD-Parameter übersteuert früher festgelegte SAME- oder SPACE-Parameter.
In der Directory-Eintragung "PLACE" wird SPREAD eingesetzt.

Shared Assign's mit unterschiedlichen PLACE-Spezifikationen:

Wird eine durch einen Prozess zugeteilte Datei von einem weiteren Prozess mit einem andern PLACE-Parameter zugeteilt, gilt für alle Prozesse die als erstes angegebene PLACE-Angabe.
Die entsprechende Directory-Eintragung bleibt durch shared Zweit-Assign's unverändert.

Empfehlung: In allen ASSIGN's für eine bestimmte Datei denselben PLACE-Parameter angeben.

Beispiele:

a) Zuteilung der Datei D-SAMMEL (3). Wenn neue Sections gebildet werden, sollen diese alle auch auf Disc 3 erstellt werden. Falls kein Platz mehr vorhanden ist: Programm abbrechen.

```
AS DS D-SAMMEL (3) SEC   ; Dir-Eintrag PLACE
                           bleibt unverändert

AS DS D-SAMMEL (3) SAME  ; Dir-Eintrag PLACE
                           wird neu = SAME
```

b) Zuteilung der Datei D-KUNDE (2). Wenn neue Sections gebildet werden, sollen diese alternierend auf alle Disc-Einheiten der GROUP, zu der Disc 2 gehört, erteilt werden.

```
AS DK D-KUNDE (2) SPR  ;   Dir-Eintrag PLACE
                           wird neu = SPEAD
```

c) Zuteilung der Datei D-ART (3). Wenn neue Sections gebildet werden, sollen diese auf derselben Disc-Einheit 3 gebildet werden. Wenn jedoch kein Platz mehr vorhanden ist, Fortsetzung auf der nächsten Disc-Einheit der GROUP, zu der Disc 3 gehört.

```
AS DA D-ART (3) SPA  ;     Dir-Eintrag PLACE
                           wird neu = SPACE
```

DELETE (Multi-Section-Dateien)

Dieses Kapitel beschreibt nur die Angaben, die für Multisection-Dateien verwendet werden.

- **Allgemeines:**

  Um eine Datei zu löschen, ist immer die Disc-Einheit ihrer ersten Section anzugeben.

- **Löschen ganzer Disc-Einheiten** mit DEL packnr (n):

  Wichtig!
  Dateien mit der ersten Section auf der betreffenden Disceinheit werden nur gelöscht, wenn der Parameter MV angegeben ist:

  DEL packnr (n) **MV**

  Bei Löschen einer ganzen Disc-Einheit werden Dateien mit der erste Section auf einer andern Disceinheit nicht gelöscht.
  Solche Sections müssen mit einem DELETE der Datei, der sie angehören, gelöscht werden.

- **Löschen einzelner Sections:**

  Wenn die Section-Folge nicht mehr lückenlos ist ("verlorene" oder "verwaiste" Sections), lässt sich eine einzelne Section mit folgendem Befehl löschen:

  DEL physname (n) **SE=sect VSN=packnr**

  wobei sect = Section-Nummer.
  packnr = Packnummer, wo sich die erste Section befindet (bzw. befinden sollte).
  Diese beiden Angaben können Sie der Directory-Ausgabe mit dem E-Parmeter entnehmen.

  Einzelne Sections intakter Dateien lassen sich nicht löschen.

DISPLAY DIRECTORY (Multi-Section-Dateien)

Dieses Kapitel beschreibt nur die Angaben, die für Multisection-Dateien verwendet werden.

- **Gesamt-Directory**

  Falls eine Disceinheit Folge-Sections von Dateien enthält, deren erste Section auf einer andern Einheit liegt, erscheint bei der Ausgabe des Gesamt-Directory die Meldung:
  VOLUME CONTAINS SECTIONS OF FILES WHICH START ON OTHER VOLUMES

  Die Ausgabe des ganzen Directory oder einer Dateigruppe mit * mit dem E- oder FU-Parameter zeigt auch die Folge-Sections von Dateien, deren erste Section auf einer andern Einheit liegt.

- **Angaben über einzelne Sections:**

  Die Ausgabe einer einzelnen Datei-Eintragung mit dem E-Parameter zeigt alle Sections der betreffenden Datei einschliesslich der Sections auf andern Disc-Einheiten.

  Die folgenden Angaben erscheinen mit dem DI-Befehl für ganze Disc-Einheiten mit dem Parameter E oder FU:

  - Jede einzelne Section.
  - Die PLACE-Angabe:
    SAME, SPREAD oder SPACE (vgl. ASSIGN)

  Die folgenden Angaben erscheinen nur beim DI-Befehl mit dem Parameter FU:

  - Die Angabe SEC1:
    Packnummer der ersten Section, falls diese auf einer andern Disc-Einheit liegt.
  - Die Angabe NEXT:
    Packnummer der nächsten Section.

COPY MOVE QBACKUP (Multi-Section-Dateien)

Dieses Kapitel beschreibt nur die Angaben, die für Multisection-Dateien verwendet werden.

Bemerkung zu $STREAM:
Multi-Volume-Dateien (Dateien, deren Sections auf mehrere Disc-Einheiten verteilt sind), lassen sich zur Zeit nicht mit $STREAM auf Streamer-Band kopieren.

KOPIEREN EINZELNER DATEIEN:

Um eine einzelne Datei zu kopieren, ist immer die Disceinheit ihrer ersten Section anzugeben.

Die Verteilung der Sections auf der Destinations-Einheit erfolgt gemäss aktueller GROUP-Definition auf der Destinations-Einheit und nach dem angegebenen Placement-Parameter.

Beim Kopieren mit COPY bleiben Anzahl un Grösse der Sections unverändert. Der Placement-Parameter wird aus der Directory-Eintragung übernommen.

Kopieren mehrerer Dateien mit COPY ...*
COPY mit Wildcard-Zeichen * (n) (m) kopiert auch die Multi-Volume-Dateien, deren erste Section auf Disceinheit (n) liegt.

MOVE SI SO FA
Zum Reorganisieren von Multi-Section-Dateien ohne Alternate-Keys ist unbedingt der Parameter FO=NEW anzugeben. Andernfalls wird eine Old-Style-Indexdatei gebildet.

KOPIEREN GANZER DISC-EINHEITEN:

WICHTIG:

Wenn **Multi-Volume-Dateien** verwendet werden, alle Discs mit dem Parameter **MV** sichern !

Wichtige Unterschiede zwischen den verschiedenen Varianten:

Ohne die Parameter MV oder MI
Multi-Volume-Dateien werden nicht kopiert.

Mit dem Parameter MV (ohne MI)
Alle Dateien deren ersten Section auf dieser Disceinheit liegt, werden kopiert.
Ihre Folge-Sections auf andern Einheiten werden "zusammengesucht" und mitkopiert.

Z.B:

```
MOV (n) (m) MV      [+ weitere Parameter]
QB (n) physband (m MT) MV
                    [+ weitere Parameter]
```

Dateien, deren erste Section auf einer andern Disceinheit liegt, werden nicht kopiert.

Mit dem Parameter MI (Mirror-Image-Kopie):

Alle Dateien und einzelnen Sections werden kopiert, unabhängig von ihrer Zugehörigkeit.
Auch das Systemfile ((GROUP)) wird kopiert.
---> MOVE mit MI nur in Sonderfällen verwenden.

Beispiele für MOVE ganzer Disc-Einheiten:

Ausgangslage:

Disc 1 enthält:		Disc 2 enthält:
Datei A, Section 1	------>	Datei A, Section 2
Datei A, Section 3	<------	
Datei B, Section 1+2		Datei S
Datei C, nur 1 Section		Datei T
Datei U, Section 2	<------	Datei U, Section 1

a) **MOVE (1) (n)** kopiert auf Disc (n):

Nur Datei B und
Datei C

---> Keine Multi-Volume-Dateien

b) **MOVE (1) (n) MV** kopiert auf Disc (n):

Datei A, Section 1, 2 und 3,
Datei B und
Datei C

---> Alle Sections der Datei A, jedoch keine Folge-Sections von Dateien mit Section Nr. 1 auf einer andern Einheit.

---> Entspricht dem Normalfall:
Alle Discs mit dem Parameter **MV** sichern.

c) **MOVE (1) (n) MI** kopiert auf Disc (n):

Datei A, Section 1 und 3,
Datei B und
Datei C und
Section 2 der Datei U

---> Alle Dateien und Einzel-Sections.
Keine Folge-Sections auf andern Einheiten.
Der vorherige Zustand von Disc n wird dabei gelöscht.
---> Vorsicht: Section 2 der Datei U ist auf der Kopie eine "verlorene" Section!

DISPLAY DIRECTORY VON QB-BAENDERN

Im Directory von Band-Kopien mit DI physband (n MT) QB werden zur Zeit die Folge-Sections nicht angezeigt.

- Notizen -

23. SYSTEM-OPTIMIERUNG

23.1 UEBERSICHT

Diese Kapitel umfasst die System-Befehle, welche der Benützer einsetzen kann, um Verarbeitungsgeschwindigkeit und System-Auslastung zu kontrollieren und zu beeinflussen.

Kontrolle der System-Auslastung:

DISPLAY STATUS USAGE:
Angaben über die momentane System-Belastung.

EX $PDP
Aufzeichnung der System-Auslastung, auch über einen wählbaren Zeitraum.

Hinweis:
Mit dem PC-Programm I-XPERT können Aufzeichnungen von $PDP auf einem ITX-WINDOWS-PC weiter analysiert und ausgewertet werden.

Beeinflussung der System-Auslastung:

SET PRIORITY
Prozess-Prioritäten ändern

LOAD, UNLOAD und DISPLAY PROGRAMS
Vorausladen von Programmen

LOAD LEVELING mit dem Befehl SET USAGE:
Setzen von Auslastungs-Limiten, bei deren Erreichen gewisse Prozesse ausgelagert oder keine neuen Prozesse mehr angenommen werden.

DISC CACHE
Verminderung der Disc-Zugriffe durch Speicherung gelesener Blöcke im Memory.

23.2 KONTROLLE DER SYSTEM-AUSLASTUNG

Die aktuelle Systemauslastung lässt sich mit den folgenden Utilities kontrollieren:

- DISPLAY STATUS USAGE
  Erweiterter Systemstatus mit Auslastungsangaben.

- EX $PDP
  Utility, das in bestimmten Zeitabständen und über einen wählbaren Zeitraum die aktuelle Systembelastung auf Bildschirm und wahlweise auf einem Drucker oder Spoolfile ausgibt.

DISPLAY STATUS USAGE

Befehls-Format:

DISPLAY STATUS USAGE [L] [TO (/ n \,/**LP** \ )]
ANY/ **DLP**/

Ausgabe-Beispiel von DI ST US:

```
SYSTEM - WIDE STATUS          MONDAY    89/08/07  12:16:35.36      ITX 6.01.14
    PROCESS   JOB-ID   CS FILE     CURR PROG  STATUS     MODE      CPU TIME     PR
   *  0.01                                      RUN    INTERACT 000:00:16.06    3
      1.02                          $PDP       IO WAIT INTERACT 000:00:19.23    1
      2.01                                     INACTIV INTERACT 000:00:19.92    3
      3.01             C            $COBOL       RUN    INTERACT 000:02:02.02    3


                      SYSTEM USAGE   MONDAY    89/08/07  12:16:36

        LOAD LEVELING THRESHOLDS:      LOW = 70   HIGH = 40
        CPU USAGE --

              OVERALL CPU USAGE:  51        EFFECTIVE CPU USAGE:  98
              TOTAL TIME SWAPPING:       000:00:05.41
              SYSTEM IDLE TIME:          000:15:57.02
              TIME SINCE SYSTEM START:   000:19:12.19

                       PROCESS     CPU USAGE(%)     SWAP-IN     SWAP-OUT
                          3.01          97              334           61
                          2.01          96               98           20
                          1.02          98               52            3
                        *  0.01         96               85            3
                        999.98          0                0            0
                        999.99         99               64            0
        MEMORY USAGE --

              TOTAL REAL MEMORY:          2004K
              SIZE OF DISK CACHE:          126 K
              USER I/O DRIVER HEAPS:       OK

              O.S. (VIRTUAL)      CODE: 2675K        DATA: 1068K
              O.S. (REAL)         CODE: 817K         DATA: 648K

                          VIRTUAL      VIRTUAL      REAL         REAL
                PROCESS   CODE(K)      DATA(K)      CODE(K)      DATA(K)
                   3.01     401          170          256          129
                   2.01       0           29            0            2
                   1.02     126           73            6           39
                 *  0.01      0           33            0           21
                 999.98      0           10            0           10
                 999.99     138          151           18           29
                           -------      -------      -------      -------
                 TOTAL       665          466          280          230


  UNIT  TYPE   VOLUME   DEVICE  SYSTEM    STATUS        I/O COUNT          ERROR
    0    DI    350000   SCSI   *SYS1,2,3*  UP             7560               0
    1    DI   REMOVED   SCSI                UP                1               0
    2    DI    350002   SCSI                UP              374               0
    3    DI   REMOVED   SCSI                UP                1               0
    4    DI    350004   SCSI                UP             2977               0
    5    DI   REMOVED   SCSI                UP                1               0

  UNIT  TYPE   STATE    DEVICE             STATUS        I/O COUNT          ERROR
                                                        TOTAL      INPUT
    0    CT  ATTACHED   7900               UP            947         82        0
    1    CT  ATTACHED   7900               UP            566         33       17
    2    CT  ATTACHED   7900               UP            427         36        0
    3    CT  ATTACHED   7900               UP            235         75        0
    4    CT  ATTACHED   7900               UP              1          0        0
    5    CT  AVAILABLE  7900               UP              3          0        0
    6    CT  AVAILABLE  7900               UP              3          0        0

  UNIT  TYPE  CODESET  DEVICE   OWNER   STATUS        I/O COUNT          ERROR
    0    LP  ASCII-64   630     0.01     UP              419               1

  UNIT  TYPE   VOLUME   DEVICE            STATUS        I/O COUNT          ERROR
    0    MT   REMOVED   6343              UP                1               0

  UNIT  TYPE   STATE    DEVICE            STATUS        I/O COUNT          ERROR
    0    LL  AVAILABLE  LLCS              UP                0               0

  UNIT  TYPE   STATE    DEVICE            STATUS        I/O COUNT          ERROR

  LOGICAL    PHYSICAL       UNIT      PROCESS OWNER STATUS KEEP NET  NODE-ID

  SNAPLOCK   SNAPLOCK       ( 0, DI)    1.02  OWN  CLOSED  NO  NO
```

Erklärung der Ausgabe von DI ST US:

LOAD LEVELING THRESHOLDS:
Dieser Teil zeigt die aktuellen Auslastungs-Limiten (vgl. Kapitel "Load leveling").

CPU USAGE --
Dieser Teil gibt Auskunft über die Prozessor-Auslastung.

OVERALL USAGE:
Aktuelle Auslastung des Prozessors in % der möglichen Voll-Auslastung.

EFFECTIVE CPU USAGE:
Anteil der Prozessor-Auslastung durch User-Prozesse in %. Die Differenz zu 100% wurde für die Memory-Verwaltung aufgewendet.

TOTAL TIME SWAPPING
Prozessorzeit, die seit dem Start für Swapping aufgewendet wurde.

SYSTEM IDLE TIME:
Seit dem Systemstart war der Prozessor während dieser Zeit inaktiv.

TIME SINCE SYSTEM START
Seit dem letzten Systemstart vergangene Zeit.

PROCESS / CPU USAGE(%) / SWAP-IN / SWAP-OUT:
Hier sind alle zur Zeit vorhandenen Prozesse aufgeführt.
Der Prozess 999.98 verwaltet den internen Bus zwischen den Subsystemen.
Der Prozess 999.99 besorgt die Prozess-Verwaltung.

CPU USAGE(%):
Gibt an, zu wievielen % die dem Prozess zugeteilten Time-Slices ausgenützt wurden. Eingeschlossen ist die Restzeit des Slice, falls die CPU-Benützung durch I/O unterbrochen wurde.
Brauchte ein Process von einem Time-Slize von 30 ms nur 27 ms, ergibt dies 90%.

SWAP-IN:
Anzahl Swap-in-Funktionen vom Swap-File ins Memory seit der Prozess-Eröffnung.

SWAP-OUT:
Anzahl Swap-out-Funktionen vom Memory ins Swap-File.
Solche Swap-Outs von Programm-Segmenten finden statt, wenn
- das Memory knapp ist oder
- wenn ein Prozess während längerer Zeit den Prozessor nicht beansprucht.

MEMORY USAGE --
Dieser Teil gibt Auskunft über die aktuelle Memory-Auslastung.

TOTAL REAL MEMORY:
Grösse des Memory. Die Differenz zum installierten Memory wird für systemliche Zwecke gebraucht.

SIZE OF DISK CACHE:
Grösse des generierten Disc-Cache-Memory (vgl. besonderen Abschnitt weiter hinten).

VIRTUAL und REAL: Jedes Programm wird beim Start ganz ins Swap-File auf dem Systemdisc geladen und von dort ins Memory. Bei Memory-Knappheit lagert das System einzelne Programm-Segmente temporär auf das Swap-File aus.
VIRTUAL = im Swap-File geladen.
Dies ist bei User-Programmen die totale Programmgrösse.
REAL = davon zur Zeit im Memory geladen.

CODE = Befehlsteil der Programme
DATA = Datenteil (Definitionen) der Programme

O.S (VIRTUAL) und O.S (REAL):
Grösse der Betriebssystem-Routinen.
Das Betriebssystem versucht möglichst viele Routinen ins REAL-Memory zu laden, der REAL-Teil wird jedoch nie so gross wie die VIRTUAL-Grösse.
Die minimale REAL-Belegung des Betriebssystems liegt bei rund 400K.

PROCESS/VIRTUAL CODE/VIRTUAL DATA/REAL CODE/REAL DATA:
Hier sind alle zur Zeit aktiven Prozesse mit der entsprechenden Memory-Belegung aufgeführt.

TOTAL:
In diesen Zahlen ist der Global-Teil von Programmen, die in mehreren Prozessen laufen, mehrmals mitgezählt, obschon er nur einmal im realen Memory gespeichert wird.

EX $PDP

Funktion:

Das Utility $PDP (Performance Diagnostic Package) dient der Kontrolle der System-Belastung. Es eignet sich besonders zur Ermittlung von Engpässen.

Start und Möglichkeiten:

Start mit

EX $PDP [/ **SC** \ ]
\ **EC** /

Parameter

SC und EC sind im Abschnitt "Data Collecton" beschrieben.

Selektion auf dem Auswahlbild:

1. SNAPSHOT:
   Ausgabe von Systembelastungs-Angaben in periodischen Abständen.

2. DATA COLLECTION:
   Definition eines oder mehrerer Zeit-Fenster (Windows), während derer die System-Belastung auf ein File auf dem System-Disc aufgezeichnet wird.

3. DATA ANALYSIS
   Ausgabe der mit DATA COLLECTION aufgezeichneten Daten auf Bildschirm oder Drucker.

4. EXIT PERFORMANCE DIAGNOSTICS PACKAGE
   Ende des Programms $PDP.

1. SNAPHOT

Funktion:

Ausgabe von Systembelastungs-Angaben in periodischen Abständen.
Die Ausgabe erfolgt immer auf dem aufrufenden Bildschirm und wahlweise auch auf einem Drucker oder Spoolfile.

Starten von SNAPSHOT:

Voraussetzungen:

- Das SHAPSHOT-Programm kann nur an einem Schirm zugleich laufen.
- Der betreffende Schirm muss Submit "Immediate" ausführen können (Submit priviledged muss erlaubt sein).
- Das Batch-Flag muss "on" sein.
  (vgl. SET BATCH ON).

Start-Befehle:

[**AS LO .....**]
Zuordnung eines Drucker oder manuellen Spoolfiles, falls Druckausgabe gewünscht wird. Der logname muss **LO** lauten.

Ein bestehendes Spoolfile kann mit OWN zugeteilt werden. Es wird darauf durch die neuen Ausgabedaten erweitert.

EX $PDP
Auswahl 1 = SNAPSHOT

SNAPSHOT erzeugt den folgenden Dialog:

< 1 > SELECT SCREEN DISPLAY FREQUENCY: (F, M, OR O)
Wahl des Zeitabstandes, in dem die Systembelastung auszugeben ist:

Eingabe	Zeitabstand bei geringer Systembelastung	Zeitabstand bei starker Systembelastung
F (frequent)	15 Sek.	1 Min.
M (moderate)	45 Sek.	1 Min. 30 Sek.
O (occasionally)	1 Min.45 Sek.	2 Min. 30 Sek.

< 2 > ARE PRINTOUTS OF SCREEN DISPLAYS DESIRED (Y OR N)
Eingabe:
Y Gewisse Bilder werden gedruckt bzw. auf das zugeodnete Spoolfile geschrieben (vgl. Frage 3).

N oder <RET> Nur Ausgabe am Bildschirm.

Falls die Frage 2 mit Y beantwortet wurde:
< 3 > SELECT PRINTOUT FREQUENCY (NUMBER BETW. 1 AND 99)
Eingabe:
Eine Zahl von 1 bis 99, die sagt, das wievielte Bild jeweils zu drucken ist, z.B:
1 = jedes Bild
2 = jedes zweite Bild
3 = jedes dritte Bild usw.

DATA OKAY (ENTER Y IF OK, Q FOR QUIT ITEM# OTHERWISE)
Eingabe:
Y = ausführen
Q = nichts ausführen und SNAPSHOT beenden.
1 bis **3** = zurück zur Frage 1, 2 oder 3 (Korrekturmöglichkeit).

Beenden von SNAPSHOT

Nach jedem Bild erscheint während einiger Sekunden die Meldung:
ENTER NEWLINE TO EXIT SNAPSHOT
Die Eingabe von **<RET>**, solange die Meldung am Bildschirm steht, beendet SNAPSHOT.
Wird die Meldung **nicht** beantwortet, folgt nach einiger Zeit ein weiteres SNAPSHOT-Bild.

Die Ausgabe von SNAPSHOT

```
                    ITX          SNAPSHOT

SNAPSHOT PROGRAM INITIATED: 09:52:27     PRINT PAGE NUMBER:          2

DATE: 87/01/14         CRT DISPLAY NUMBER:          3

   CPU POWER USAGE          REAL MEMORY USAGE     INPUT/OUTPUT
   ---------------          -----------------     ------------
99%                      99%
95%                      95%                     SINCE SYSTEM RESET:
90%                      90%                           TOTAL
85%                      85%  **********         DI  =     32486
80%                      80%  **********         CRT =     27460
75%                      75%  **********         LP  =         0
70%                      70%  **********         MT  =         0
65%                      65%  **********
60%  **********          60%  **********         SINCE LAST SCREEN:
55%  **********          55%  **********               TOTAL    PER SECOND
50%  **********          50%  **********         DI  =       253    12.65
45%  **********          45%  **********         CRT =       236    11.80
40%  **********          40%  **********         LP  =         0     0.00
35%  **********          35%  **********         MT  =         0     0.00
30%  **********          30%  **********
25%  **********          25%  **********         MEMORY: TOTAL REAL =  3020K,
20%  **********          20%  **********                 OVERHEAD = MINIMAL
15%  **********          15%  **********         TIME THIS SCREEN: 09:53:51
10%  **********          10%  **********         TIME LAST SCREEN: 09:53:31
05%  **********          05%  **********         TERMINALS ATTACHED:   26
00%  **********          00%  **********         ACTIVE PROCESSES:  23
```

Die 3 Titelzeilen erscheinen nur in der Druckausgabe.

PRINT PAGE NUMBER = Seitennummer der Liste.

CRT DISPLAY NUMBER = Nummer des Bildes aus demselben Lauf von SNAPSHOT, z.B:
Wenn jedes 2. Bild gedruckt wurde, erscheinen die Page Numbers 1, 3, 5 usw.

CPU POWER USAGE
Prozessorauslastung in % der Vollauslastung seit dem letzten SNAPSHOT-Bild.

REAL MEMORY USAGE
Memory-Auslastung in % der Vollauslastung zum Zeitpunkt des SNAPSHOT-Bildes.

Die Ausgabe von SNAPSHOT (Fortsetzung)

INPUT/OUTPUT

Anzahl physische Ein-/Ausgabe-Operationen, aufgeteilt nach:
SINCE SYSTEM RESET:
Seit dem Start bzw. Errorlog-Ausgabe.
SINCE LAST SCREEN:
Seit dem letzten SNAPSHOT-Bild.
Die I/O-Operationen sind aufgeteilt nach:
DI = Disc
CRT= Bildschirm
LP = Drucker
MT = Magnetband

MEMORY

TOTAL REAL = Memorygrösse, ohne Firmware.
OVERHEAD = Systembelastung durch Memory-Verwaltung:
NONE = keine
MINIMAL = sehr klein
MODERATE = mittel
HEAVY = stark

TIME THIS SCREEN: Ausgabe-Zeit dieses Bildes
TIME LAST SCREEN: Ausgabe-Zeit des letzten Bildes.

TERMINALS ATTACHED: Anzahl attached Bildschirme.

PROCESSES ACTIVE: Anzahl zur Zeit aktive Prozesse.

Wichtige Bemerkungen zu SNAPSHOT:

SNAPSHOT bildet für jedes Bild einen Submit-Batch-Job mit dem Jobnamen DISTUSUB. Dieser läuft mit der bestmöglichen Priorität, die gemäss Sysgen für Submit möglich ist; wenn möglich mit Priorität 1.
Dies ergibt eine gewisse System-Belastung und erzeugt Meldungen am Firmware-Bildschirm (Schirm 0).

Wenn das SNAPSHOT-Programm mit hoher Frequenz (FREQUENCY = F) läuft, kann es das System erheblich belasten.

Das SNAPSHOT-Programm bildet automatisch folgende Arbeits-Dateien auf Disc (SYS3):

SNAPLOCK, 1 Sektor
DISTUSF
SUBDISTF

Diese Dateien werden von SNAPSHOT normalerweise gelöscht. Je nach dem Zeitpunkt des Programm-Abbruchs können sich jedoch DISTUSF und SUBDISTF nach Programmende noch auf dem Disc befinden.
Bei einem späteren Start von SNAPSHOT werden diese Dateien automatisch gelöscht.

2. DATA COLLECTION UND DATA ANALYSIS

DATA COLLECTION:
Definition eines oder mehrerer Zeit-Fenster (Windows), während derer die System-Belastung auf ein File PDPDATACOL auf dem System-Disc aufgezeichnet wird.

DATA ANALYSIS
Nach Beendigung der DATA COLLECTION:
Ausgabe der aufgezeichneten Daten auf Bildschirm oder Drucker. Diese Ausgabe ist auch mehrmals mit verschiedener Selektion ausführbar.

Die Beschreibung dieser Funktionen beschränkt sich auf die wichtigsten Punkte.
Weitere Informationen erscheinen beim Aufruf am Bildschirm. Eine auführliche Beschreibung finden Sie im englischen Original-Reference Manual.

DATA COLLECTION

Definition eines oder mehrerer Zeit-Fenster (Windows), während derer die System-Belastung auf ein File auf dem System-Disc aufgezeichnet wird.
Innerhalb jedes Windows ist eine Periode (Frequency) in Minuten anzugeben. Alle Angaben werden pro Periode aufgezeichnet.
Die Zeiten können jeweils auf 5 Minuten angegeben werden (z.B: 10:00 oder 10:05 usw). Die kleinste Periode ist 5 Minuten.

Start:

Falls ein eigenes Collection File gewünscht wird:
AS PDPCOLLECT physname (n) NE grösse SEC

Da diese Datei kann sehr gross werden kann, ist Multi-Section sinnvoll (Parameter SEC).
Ohne eine Datei-Zuteilung wird automatisch ein File PDPDATACOL/ (neueste Generation) auf Disc (SYS3) erstellt.

EX $PDP

Wahl: 2. DATA COLLECTION

Selektionen:

Mit der Funktion 1 (ADD A WINDOW) lassen sich Zeit-Bereiche für die Aufzeichnung definieren.

Beispiel:

Window	Start	End	Frequency
1	13:20	14:00	10

Von 13:20 bis 14:00 wird die System-Belastung in 10-Minuten-Perioden aufgezeichnet. (Doppelpunkte bei der Eingabe der Zeit auch eintippen!)

DATA COLLECTION (Fortsetzung)

Mit der Funktion 5 (DATA TO BE COLLECTED) kann genauer bestimmt werden, welche Daten zu sammeln sind.
Hinweis:
Die Daten-Kategorien "Disk I/O Timings" sollte nur für Multibus-Systeme und nur für Auswertungen mit I-XPERT auf ON gesetzt werden, um Disc-Platz zu sparen!

Mit der Funktion 6 (PROCEED WITH DATA COLLECTION) wird die Aufzeichnung der Daten gestartet.

Daten-Aufzeichnung:
Nach der Wahl "Proceed with data collection" werden die Messdaten durch einen SPAWN-Prozess mit dem Programm $PDP auf die Collection Datei aufgezeichnet.
Die Data Collection läuft automatisch ab, ohne Bediener-Eingriffe.

Dieser Spawn-Prozess verschwindet, sobald die angegebene Zeispanne (Window) abgelaufen ist.
Er kann auch mit ABORT xxx.yy vorzeitig abgebrochen werden.

Starten der Data Collection ohne Dialog

Start Data Collection:

EX $PDP **SC** [minuten [tage]]

Startet die Data Collection automatisch.
minuten = Länge einer Messperiode, mind. 3.
Ohne diese Angabe dauert die Messperiode bis zur Beendigung mit EX $PDP EC bzw. bis Mitternacht.
tage = Anzahl Tage, wo jeweils um dieselbe Zeit die Data Collection automatisch gestartet werden soll.

End Data Collection:

EX $PDP **EC** [minuten]

Stoppt die Data Collection.
minuten = Zeitdauer in Minuten, bis die Data Collection gestoppt wird.
Während dieser Zeit ist der laufende Prozess blockiert.
Ohne diese Angabe wird "sofort" angenommen.

Der Befehl EX $PDP mit den Parametern SC und EC kann auch in Controlstrings stehen.
Damit kann die Data Collection ohne Benützer-Dialog gestartet und beendet werden.

Data Analysis ist weiterhin im Dialog auszuführen.

DATA ANALYSIS

Diese Phase wird gestartet, nachdem die Data-Collection-Aufzeichnungen abgeschlossen sind.

Sie kann erst ausgeführt werden, nachdem das zu analysierende Window abgelaufen ist.

Start:

Falls ein eigenes Collection File zu verarbeiten ist:
AS PDPANALYZE physname (n)

Ohne eine Datei-Zuteilung folgt später eine Frage nach dem zu verarbeitenden File.

EX $PDP

Wahl: 3. DATA ANALYSIS

Selektionen:

CHOOSE FILE GENERATION NUMBER:
- Eingabe der Generationsnummer des Files DPDATACOL(SYS3), / = neueste Gen.
  oder:
- Eingabe <RET>:
  Es folgt die Frage:
  ENTER PDP DATA COLLECTION FILE AND UNIT
  Hier den Physnamen und die Disc-Einheit eines beliebigen Data Collection Files eingeben.

DATA ANALYSIS (Fortsetzung)

Auf einem weiteren Bild ist wählbar:

1. CHANGE ANALYSIS TIME WINDOW und
2. CHANGE ANALYSIS DATE:
   Zeitraum (von / bis)

3. CHANGE REPORT NAME:
   Collection-Datei, die zu analysieren ist.

4. GO TO CRT DISPLAY OPTIONS:
   Zum Auswahlbild für Ausgabe am Bildschirm.
5. GO TO PRINTER REPORT OPTIONS:
   Zum Auswahlbild für Ausgabe am Drucker.

6. GO TO I-XPERT FILE CREATION:
   Zum Erstellen einer Datei, die später mit I-XPERT an einem PC ausgewertet wird.
   Physname/Generation und Disc-Unit des erstellten I-XPERT-Files wird gemeldet mit:

   PDP313 DATA ANALYSIS COMPLETE.
   FILE CREATED: XPERTRPT/gen(n)

   ---> Physname/gen und Disc-Einheit notieren !

In den "Report Options" kann selektiert werden, welche Angaben auszugeben sind,
z.B:
- System-Engpässe = Perioden mit hoher Prozessor- oder Disc-Belastung
- Anzahl I/O's pro Disc-Controller oder Disc-Drive
- Anzahl I/O's für verschiedene Geräte-Arten
- Swapping-Summary

Für Druck-Ausgabe geht die Selektion noch weiter:
- Prozessor-Auslastung (CPU) pro Prozess
- I/O-Activity auf Files, pro Prozess
- I/O-Activity auf Cached Files
- I/O-Activity auf Remote Files (via ITXNET), pro Prozess
- I/O pro Terminal
  usw.

$PDP, DATA ANALYSIS (Fortsetzung)

Zu beachten:

Die durch $PDP gebildeten Files müssen nach beendeter Data Analysis durch den Benützer gelöscht werden:

Das Data Collection File:
PDPDATACOL (SYS3) - alle Generationen!
oder
eigene Collection Files.

Solange diese Files nicht gelöscht sind, können die Aufzeichnungen mit anderer Selektion beliebig oft analysiert werden.

Eventuell andere eigenen Files
(z.B I-XPERT-Files)

23.3 PROZESS-PRIORITAETEN

Die Bedienung der Prozesse durch den zentralen Processor wird nach dem Timesharing-Prinzip mit Prioritäts-Stufen geregelt.

Einem Prozess kann eine von acht Prioritäten zugeteilt werden. Solange ein Prozess hoher Priorität (z.B. 1) Processorzeit benötigt, wird die Bedienung aller Prozesse mit niedrigeren Prioritäten zurückgestellt. Innerhalb derselben Prioritätsstufe gilt die gleiche Rangordnung der Prozesse (Round-robin-Prinzip). Der Time-Slice, d.h. die Zeit, während der ein Prozess den Processor benützen darf, bis er wieder hinten an der Warteschlange seiner Prioritätsstufe plaziert wird, beträgt 30 ms.

Beispiel:

Priorität	Prozesse	
1	001.01	003.01
2	003.03	
3	---	
4	001.02	002.02

Prozess 001.01 wird als erster bedient. Wenn der Processor durch eine Ein/Ausgabe-Funktion oder durch Ablauf des Time-Slices frei wird, kommt Prozess 003.01 an die Reihe. Falls auch dieser keine Processorbedienung verlangt, wird Prozess 003.03 bedient usw. Die nicht voll ausgenützte Restzeit eines Time-Slices verfällt, wenn ein aktiver Prozess durch eine Ein/Ausgabe-Funktion unterbrochen wird.

Im Normalfall werden interaktiven Prozessen die Prioritäten 1 bis 3, Batch-Prozessen die Priorität 8 zugeteilt.

Prioritäten lassen sich mit dem System-Befehl DISPLAY STATUS jederzeit überprüfen (siehe Kapitel "System-Befehle).
Mit dem System-Befehl SET PRIORITY lassen sich die Prioritäten jederzeit verändern.

SET PRIORITY

Format:

```
SET PRIORITY [ /xx.yy\ ] TO  p
             \ .yy /
```

Parameter:

xx.yy	: Irgendein Prozess
.yy	: Prozess-Nr. .yy des eigenen Schirms
keine xx.yy-Angabe	: Der eigene, laufende Prozess.
p	: Die neue Priorität des Prozesses. Von 1 (beste Priorität) bis 8.

Bemerkungen:

Jede Prioritätsänderung wird mit der folgenden Meldung bestätigt:

```
PROCESS xx.yy NOW EXECUTING WITH PRIORITY p
```

Die Priorität bleibt für die Dauer des Prozesses (gleiche Prozess-Identifikation) erhalten.

Vorsicht bei SUBMIT-Batch-Prozessen, Compilationen, Sort-Läufen und Berechnungs-Programmen:
Laufen solche Verarbeitungen mit bevorzugter Priorität ab, können die übrigen Arbeiten stark gebremst werden!

23.4 VORAUSLADEN VON PROGRAMMEN

Befehle:

LOAD	Voraus-Laden von Programmen. Beschleunigt den EXECUTE-Befehl solcher Programme und vereinfacht die Memory-Aufteilung.
UNLOAD	Entladen vorausgeladener Programme
DISPLAY PROGRAMS	Kontrolle, welche Programme mit LOAD geladen sind.

LOAD

Funktion:

Voraus-Laden von Programmen.
Später ausgeführte EX-Befehle für die betreffenden Programme müssen diese nicht mehr im Disc-Directory suchen und ins Memory laden.

Format:

LOAD **physname** (n,DI)

physname	Muss ein Objekt-Programm sein. Auch für System-Utilities (Mit $ beginnende Namen) anwendbar. Für System-Utilities wird jede Angabe der Disc-Einheit ignoriert. Um eine andere Generation als 1 zu laden, ist auch eine Generationsangabe /nnn möglich.

Bemerkungen:

Ein LOAD-Befehl für ein Programm gilt für sämtliche Prozesse des Systems. Für ein bestimmtes Programm muss deshalb nur einmal ein LOAD-Befehl in irgendeinem Prozess ablaufen.

Durch Vorausladen mit LOAD werden spätere EXECUTE-Befehle beschleunigt.

Vorsicht:
Die Disc-Angabe in EXECUTE-Befehlen für vorausgeladene Programme werden dabei ignoriert!

Vorausgeladene Programme belegen internes Memory. Sie werden jedoch bei Memory-Knappheit vom System swapped, falls sie nicht benützt (executing) sind.

Mit dem Befehl DI PR lässt sich kontrollieren, welche Programme vorausgeladen sind.

Nicht mehr im Memory benötigte vorausgeladene Programme werden mit dem System-Befehl UNLOAD entladen. Spätestens beim System-Abschluss ist je ein UNLOAD-Befehl für jedes Programm erforderlich, das noch im DI PR-Verzeichnis erscheint.

Programme, die mit LOAD geladen sind, können mit dem COBOL-Befehl CALL auch von Programmen auf andern Disc-Einheiten aufgerufen werden.

Ist der LOAD-Befehl in einem Control-String und das betreffende Programm bereits mit LOAD geladen, erfolgt nur eine Meldung und der Control-String läuft weiter. Dies erlaubt "vorsorgliche" LOAD in verschiedenen Abläufen.

Beispiele:

a) Benützer-Programm:

```
LO P-ERFASSEN (3)
```

b) System-Utility:

```
LO $EDIT
```

UNLOAD

Funktion:

Entladen eines Programms, das zuvor mit dem Befehl LOAD vorausgeladen wurde.

Format:

UNLOAD **physname**

physname	Name eines mit LOAD vorausgeladenen Programms oder System-Utilities, ohne Generationsnummer.

Bemerkungen:

Ein Programm, das sich in irgendeinem Prozess in Ausführung befindet, kann nicht mit UNLOAD entladen werden.

Vor dem REMOVE der Disc-Einheit, von der das Programm geladen wurde, oder vor dem STOP-Befehl müssen alle vorausgeladenen Programme mit UNLOAD entladen werden.

Ist ein UNLOAD-Befehl in einem Control-String und das betreffende Programm nicht mit LOAD geladen, erfolgt nur eine Meldung und der Control-String läuft weiter.

Beispiel:

Die Programme P-500 und $EDIT entladen:

```
UNL P-500
UNL $EDIT
```

DISPLAY PROGRAMS

Funktion:

Alle mit dem System-Befehl LOAD vorausgeladenen Programme werden mit Namen, Generation und Disc-Einheit ausgegeben.
(Vergleiche System-Befehl LOAD)

Format:

```
DISPLAY PROGRAMS  [ TO ( / n \,/LP \ ) ]
                       \ANY/ \DLP/
```

Beispiel:

DI PR

ergibt im Fall der mit LOAD geladenen Programme
MENU510-D(4) uns LOGEND(5):

```
-------------------------------------------------------
| PRELOADED PROGRAMS      TUESDAY 89/08/10  11:44:30  |
| PROGRAM    GEN  UNIT                                |
| MENU510-D  001    4                                 |
| LOGEND     001    5                                 |
-------------------------------------------------------
```

23.5 LOAD LEVELING

Prinzip:

Das System prüft alle 30 Sekunden, wie stark es mit Ein- und Auslagern von Programmteilen beschäftigt ist (Swapping).
Erreichen diese System-internen Operationen einen zu grossen Anteil an der System-Leistung, wird die Eröffnung neuer Prozesse gestoppt oder inaktive Prozesse temporär aus dem Memory ausgelagert.

Trash levels:

Low Level (in %):
Stehen nur noch soviele % der Prozessor-Leistung für die Verarbeitung zu verfügung, nimmt das System keine neuen Prozesse mehr an.

High Level(in %):
Stehen nur noch soviele % der Prozessor-Leistung für die Verarbeitung zu verfügung, werden Background-Jobs und Prozesse, die seit über 10 Minuten inaktiv sind, ausgelagert. Dadurch wird mehr Memory für die übrigen Prozesse frei.

Wird an einem Bildschirm während längerer Zeit keine Eingabe gemacht, erfolgt die Auslagerung des betreffenden Prozesses automatisch. Mit der nächsten Eingabe wird der Prozess wiederum automatisch aktiviert.
In beiden Fällen erscheint auf dem Schirm 0 eine "ROLLED OUT"- bzw. "ROLLED IN"-Meldung.

Die gebräuchlichen THRASH LEVEL-Werte sind von der System-Grösse abhängig.

Beispiel:

Low = 50 %	(d.h. wenn nur noch 50 % der System-auslastung für User-Programme zur Verfügung steht).
High = 30 %	(d.h. wenn nur noch 30 % der System-auslastung für User-Programme zur Verfügung steht).

Kontrolle und Aenderung der Thrash Levels:

Zur Kontrolle der Thrash Levels und der übrigen System-Auslastung wird der System-Befehl DISPLAY STATUS USAGE verwendet.

Die beiden Thrash Levels lassen sich mit dem System-Befehl SET USAGE verändern.

SET USAGE

Format:

```
SET USAGE  /L=nn\  [ /H=nn\ ]
           \H=nn/    \L=nn/
```

Beschreibung:

Mindestens einer der beiden Parameter L=nn oder H=nn ist erforderlich.

nn kann 00 bis 99 sein und bezeichnet den Prozent-Anteil der System-Leistung, der neben dem Swappen für Verarbeitungen verbleibt.

L = nn : Low level. Ist diese Auslastung erreicht, können keine neuen Prozesse mehr eröffnet werden.

H = nn : High level. Darf nicht grösser als der low level sein.
Beim Erreichen dieser Limite werden Submit-Batch- und inaktive Prozesse "rolled out".

Bemerkungen:

Die Auslastungs-Limiten können mit dem Befehl DI ST US kontrolliert werden.

Die Auslastungslimiten bleiben bis zu einem weiteren SET USAGE-Befehl oder bis zum System-Abschluss erhalten.

Beispiele für SET USAGE:

a) Den Low-level auf 60, den High-level auf 10 % der Systemleistung setzen:

```
SE US  L=60  H=10
```

b) Nur den Low-level auf 50 % verändern

```
SE US  L=50
```

23.6 DISC-CACHE

Disc-Cache ist eine Technik, um die Anzahl physischer Disc-Zugriffe zu reduzieren.

Bei der System-Generierung kann ein Teil des Memory als "Disc Cache Memory" reserviert werden (64 KB bis 50 % des Memory).
Alle gelesenen Blöcke der Dateien, die mit dem Cache-Parameter assigned sind, werden im Cache Memory gespeichert. Soll ein solcher Block später wieder gelesen werden, holt ihn das System aus dem Memory, ohne einen Disc-Zugriff auszuführen.
Schreibfunktionen werden auf Disc und im Cache-Memory ausgeführt. Dadurch sind die Dateien auch im Falle eines Absturzes auf dem aktuellen Stand.
Bei vollem Cache-Memory werden die am wenigsten oft gelesenen und die längsten nicht mehr gelesenen Blöcke durch neuere überdeckt.

Wichtige Bemerkungen:

Disc-Cache bringt nur weniger Disc-Zugriffe und Zeitersparnis, wenn
- dieselben Blöcke eines Files
- oft gelesen werden.

Für Disc-Cache geeignet sind somit vor allem:
- Index-Dateien (viele Zugriffe auf den Index)
- Dateien, die gleichzeitig von mehreren Prozessen bearbeitet werden.
- Dateien die von Batch-Programmen im Direktzugriff bearbeitet werden.

Das Disc-Cache-Memory muss für die zu speichernden Blöcke gross genug definiert sein, damit wiederholt gelesene Blöcke nicht durch andere Blöcke überdeckt werden, bevor sie wieder gelesen werden.
Der gröbste Index von Index-Dateien wird auch ohne Cache-Zuteilung im Memory behalten.

Für Dateien, wo die Bedingung "dieselben Blöcke werden oft gelesen" nicht erfüllt ist, sollten nicht mit dem CA-Parameter zugeteilt werden. Ihre Blöcke würden den Cache-Memory-Bereich unnötig füllen.

Aktivierung des Disc-Cache-Mechanismus:

Der **ASSIGN**-Befehl für die betreffenden Dateien muss den Parameter **CA** enthalten,

z.B: AS KUNDE D-KUNDE (2) **CA**

Achtung:
Der erste Assign auf dieses File bestimmt, ob Cache aktiviert wird oder nicht.

Ist das File ohne CA schon assigned, ergeben Assigns mit CA die folgende Meldung:
Z733 DISK CACHE MISMATCH-DEFAULT TO ASSIGNMENT USING POOL
und der CA-Paramter wird ignoriert.

Deshalb empfiehlt es sich, alle Assign's der betreffenden Dateien mit dem Parameter CA auszuführen, damit die Reihenfolge der ablaufenden Assigns keine Rolle mehr spielt.
Varianten:
Siehe unten oder auf der folgenden Seite!

Disc Cache austesten, ohne CA in jedem Assign, Variante 1:

An einem Bildschirm, der während der Test-Zeit nie detached und nie mit RET oder BY abgemeldet wird:
Das File, dessen Blöcke zu speichern sind, zuteilen mit

AS logname-x physname (n) **CA KE**

Als lognamen-x einen Namen verwenden, der sonst nie verwendet wird.

Diese Zuteilung mit Cache bleibt nun solange aktiv, wie der Prozess besteht.
Alle Zuteilungen des betreffenden Files werden in dieser Zeit stillschweigend mit Cache angenommen.
OWN-Zuteilungen anderer Prozesse sind jedoch nicht möglich!

Cache-Zuteilungen aufheben:

Den Prozess beenden, z.B. mit RET, BY oder Break und A.

Disc Cache aktivieren, ohne CA in jedem Assign, Variante 2:

Einen Controlstring erstellen:

```
AS logname-x physname-1 (n) CA KE RE
AS logname-y physname-2 (n) CA KE RE
 ; usw. für alle Dateien, für die CA gelten soll
EQ PROZESS #VALUE(#PID) / 100
SUS PROZESS        ;Suspendiert sich selbst
```

Als lognamen-x und -y sind irgendwelche lognamen möglich.

Diesen Controlstring mit SUBMIT starten:

Er teilt sich alle darin aufgeführten Files mit CA zu und supendiert sich darauf selbst.
Bemerkung:
Diesen Controlstring starten, bevor andere Submit-Jobs desselben Terminals gestartet werden, weil er auch alle andern Submit-Jobs des eigenen Schirmes supendieren würde.

Diese Zuteilungen mit Cache bleibt nun solange aktiv, wie der Prozess besteht.
Alle Zuteilungen des betreffenden Files werden in dieser Zeit stillschweigend mit Cache angenommen.
OWN-Zuteilungen anderer Prozesse sind jedoch nicht möglich!

Cache-Zuteilungen aufheben:

Den obigen Submit-Job abbrechen mit:

```
ABORT J=jobname    oder
ABORT xxx.yy
```

- Notizen -

24. START UND ABSCHLUSS VON TAM

EINSATZ VON TAM

TAM (Terminal Access Method) ist die Grundsoftware für verschiedene Kommunikations-Prozeduren, z.B:

- RBS (Remote Batch Kommunikation)
- ITX-NET über eine DLC-ABM oder X.25-Verbindung
- Programm-zu-Programm-Kommunikation

u.a.m.

Die TAM-Software muss geladen werden, bevor die betreffende Kommunikations-Software aktiviert werden kann.
Vor dem System-Abschluss ist die TAM-Software wieder zu entladen.

Dieses Kapitel beschreibt nur das Starten und den Abschluss von TAM.
Weitere Informationen zu TAM und den verschiedenen Kommunikations-Systeme werden in besonderen Dokumentationen beschrieben.

Die hier beschriebenen Befehlsfolgen können auch im Rahmen eines Control-Strings ablaufen.

TAM STARTEN

Bevor die eigentliche Kommunikations-Software gestartet werden kann, ist TAM mit dem Systemprogramm $TCM zu laden.
Die TAM-Software bleibt geladen, bis sie wiederum mit EX $TCM entladen wird.

```
AS logname physname (n) SH
EX $TCM
GO TAM logname
QUIT
```

logname: logischer Name für das EDF-Objekt-File, frei wählbar.
Meistens wird A verwendet.

physname: Name des EDF-Objekt-Files, das die Uebermittlungs-Netzwerk-Definition enthält.

Diese Prozedur eröffnet einen Spawn-Prozess im System mit dem Systemprogramm $TAMDM.

TAM ABSCHLIESSEN

Vor dem System-Abschluss muss das TAM-Subsystem entladen werden:

```
EX   $TCM
STOP TAM
QUIT
```

Damit verschwindet der Spawn-Prozess des TAM aus dem System.

25. DAS HARDWARE-FEHLERPROTOKOLL (ERROR-LOG)

KURZBESCHREIBUNG

Diese Liste dient dem Techniker und dem NCR-System-spezialisten als Hinweis auf notwendige Unterhaltsarbeiten und zum Auffinden von Störungsursachen.

Das Fehlerprotokoll wird vom Betriebssystem automatisch auf dem System-Disc gespeichert. Der dazu bestimmte Bereich ist im Disc-Directory als ((ERRLOG)) aufgeführt.

Bevor das Protokoll auf dem System-Disc voll ist, gibt das Betriebssystem am Bildschirm eine Warnung aus:

X004 ERROR LOG HAS x SECTORS BEFORE WRAPPING -
WRAP-COUNT = y

x = Noch freie Sektoren im Protokoll-File;
y = Zähler, der angibt, wie oft das Protokoll-File seit dem letzten Ausdruck schon überschrieben wurde

Ist das Protokoll **voll**, erfolgt die Meldung:

X005 ERROR LOG WRAPPING - WRAP COUNT = y

Das Protokoll-File wird **von vorn** her **überschrieben**; die ältesten Eintragungen gehen dabei **verloren**!

AUSDRUCKEN / WEGKOPIEREN / LOESCHEN DES ERROR-LOG

Dazu wird eine Arbeitsdatei benötigt, die nach dem Ausdrucken gelöscht werden kann.

Benötigte System-Befehle:

AS EO physname (n) / NE sekt PR [AP] [SC] \
\ OW /

AS LO .... Drucker oder Spoolfile

EX $LOGUTIL [,RE] [,EX] [,I]

physname: frei wählbar.

sekt: Mindestens 1.25 x soviele Sektoren, wie das File ((ERRLOG)) auf dem System-Disc umfasst.

RE: Das Errorlog auf dem Systemdisc wird nach dem Ausdrucken gelöscht.

EX: Das File physname wird durch die neuen Errorlog-Daten erweitert. Die darauf bestehenden Daten bleiben erhalten.

I: erlaubt die Selektion für **spezifische** Auswertungen des Errorlog. Vergleiche auch unter "Beispiele, Sonderfall".
Darüber hinaus ist diese Variante vor allem für Systemspezialisten und Techniker bestimmt.
Falls diese Variante **irrtümlich** gewählt wurde, kann das Programm mit der Eingabe **DONE** bzw. **FINISH** beendet werden, je nach Bild.

Bemerkung

Da System-Fehler während des Ablaufs von $LOGUTIL nicht ins Protokoll eingetragen werden, sollte $LOGUTIL möglichst allein am System oder zumindest mit bevorzugter Priorität ablaufen.

Beispiele:

a) Das ganze Errorlog ausdrucken, ohne es auf dem Systemdisc zu löschen. Die Arbeitsdatei ist neu auf Disc 1 zu eröffnen:

```
AS EO ERRLOG (1) NE 400 PR AP
AS LO (LP)
EX $LOGUTIL
```

b) Das Errorlog ist nach dem Ausdrucken auf dem Systemdisc zu löschen. Die bestehende Arbeitsdatei ERRLOG auf Disc 1 kann benützt (überschrieben) werden:

```
AS EO ERRLOG (1) OW
AS LO (LP)
EX $LOGUTIL RE
```

c) Wie Beispiel b). Die bestehende Arbeitsdatei ERRLOG aus einem früheren Lauf von $LOGUTIL ist jedoch durch die neuen Daten zu erweitern und ihr alter und neuer Inhalt zusammen auszudrucken.

```
AS EO ERRLOG (1) OW
AS LO (LP)
EX $LOGUTIL EX RE
```

d) Sonderfall:

Das Errorlog auf eine Arbeitsdatei schreiben und auf dem Systemdisc löschen:

```
AS EO ERRLOG (1) NE 200 AP
EX $LOGUTIL,I
```

Die Meldung U203 ENTER PROCESSING OPTION
mit **B** beantworten.
Die Meldung
U206 DO YOU WANT TO RESET THE ERROLOG FILE
mit **Y** beantworten.

Das oben aufbereitete File ausdrucken:

```
AS EO ERRLOG (1)
AS LO (LP)
EX $LOGUTIL,I
```

Die Meldung U203 ENTER PROCESSING OPTION
mit **C** beantworten.
Die Meldung U201 ENTER OPTION (1-12, MENU)
mit **1** beantworten.
Die Meldung
U205 DO YOU WANT TO DISPLAY THE PARAMETER MENU
mit **Y** beantworten.

Nun erscheint ein Auswahlbild mit
der Meldung U200 ENTER N/L WHEN NO CHANGES.

Hier sind folgende Eingaben zu machen:

1	Der Cursor springt auf die erste Zeile,
N	Der Cursor springt auf die Meldung U200 ENTER N/L WHEN NO CHANGES,
2	Der Cursor springt auf die zweite Zeile,
Y	Der Cursor springt auf die Meldung U200 ENTER N/L WHEN NO CHANGES.

<RET> eingeben.

Nach dem Ausdruck die Meldung
U201 ENTER OPTION (1-12, MENU)
mit **12** beantworten.

Darauf kann des File ERRLOG gelöscht werden.

26. UNTERHALT DES SECURITY-SYSTEMS ($ACCESS)

26.1 UMFANG DES ZUGRIFFS-SCHUTZES:

Bei Aktivierung eines Bildschirms (SYSTEM READY FOR LOG-ON) wird der betreffende Schirm nur benützbar, wenn eine gültige Kombination von User-Id und Passwort eingegeben wird.
Nach dem Drücken der Break-Taste erscheinen die Aufforderungen:

ENTER USER ID:
ENTER PASSWORD:

VARIANTEN:

Sicherheits-Stufen:
ITX erlaubt eine der folgenden Sicherheitsstufen, welche für das ganze System gilt:

- Keine User-Id/Passwort-Eingaben.
  In diesem Fall wird das Programm $ACCESS nicht benützt.
- Nur die User-Id wird verlangt.
- User-Id und dazugehöriges Passwort werden verlangt.

User-Id und Passwort sind nicht an bestimmte Bildschirme gebunden. Mit jeder gültigen Eingabe kann irgendein Schirm aktiviert werden.

Reaktion des Systems auf ungültige User-Id/Passwort:
Bei Fehl-Eingaben ist wählbar:

- Unbeschränkte Wiederholungsmöglichkeit.
- Eingabesperre von wählbarer Dauer nach einer bestimmten Anzahl Fehleingaben bis zum nächsten Versuch.
- Automatischer Detach des Bildschirms nach einer bestimmten Anzahl Fehleingaben.

User-Typen:
$ACCESS kennt 2 User-Typen:

- Administrator: Kann alle Unterhaltsfunktionen des Programms $ACCESS ausführen.
  Mindestens 1 User muss vom Typ Administrator sein.
- Normal-User: Kann als einzige Unterhaltsfunktion sein eigenes Passwort verändern. Bei Bedarf lässt sich $ACCESS für Normal-User auch ganz sperren.

User-spezifische Start-Controlstrings:
Bei Bedarf kann für jeden User ein Controlstring festgelegt werden, der immer automatisch abläuft, wenn sich der User anmeldet.

Wichtiger Hinweis:
---> Passwörter werden nirgends dokumentiert. Sie
---> bleiben auch für den Administrator unsichtbar !

Was kann man tun, wenn ein Passwort vergessen wurde?
Den User löschen und mit einem neuen Passwort neu erfassen (vgl. unter "User-Definition").

26.2 KONZEPT DES PROGRAMMS $ACCESS

Alle Security-Angaben werden mit diesem Utility unterhalten.
Das Programm kann im Menu-Modus (Dialog mit Auswahlbildern) oder im Befehls-Modus (Befehlsfolge, z.B. im Rahmen eines Controlstrings) betrieben werden.

Start des Programms:

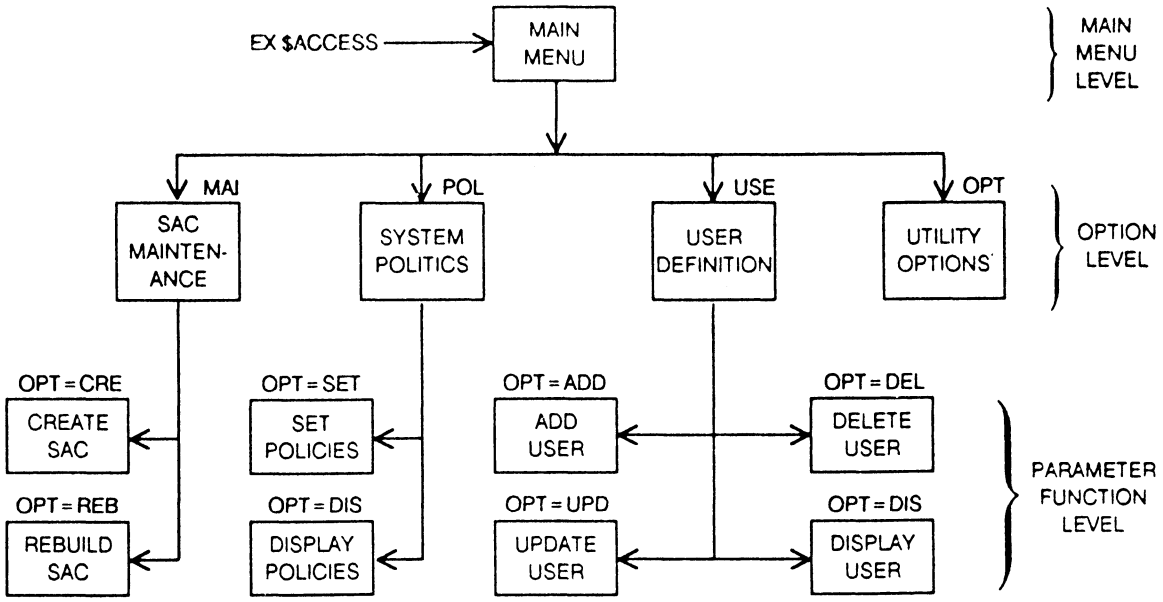
```
EX $ACCESS
```

Das Programm enthält folgende Haupt-Funktionen:

- Utility Options: Wahl des Menu- oder des Befehls-Modus.
- User-Definition: Unterhalt von User-Identifikationen und dazugehörenden Passwörtern, Festlegen des User-Typs.
- SAC Maintenance: Erstmaliger Neuaufbau und spätere Reorganisation des System-Files ((SAC)), wo alle Security-Angaben gespeichert sind.
- System Policy: Festlegen diverser Parameter, z.B:
  Ist ein Passwort erforderlich?
  Minimale Passwortlänge?
  System-Reaktion auf Fehl-Eingaben?
  Kann ein Normal-User seine eigenen Passwörter ändern?

Die folgenden Erläuterungen gelten in erster Linie für den Menu-Dialog-Modus. Einen besonderen Abschnitt über den Befehls-Modus finden Sie am Ende des Kapitels.

Funktions- und Bild-Hierarchie in $ACCESS



Beispiel eines Bildes und allgemeine Eingabe-Regeln:

```
                              DISPLAY USERS                              3.7

                          ENTER PARAMETER CODE:
+-------------------------------------------------------------------------+
: (1) TYPE OF DISPLAY     = ALL                                           :
: (2) USER ID             = NONE          (3) NODE ID           = LOCAL   :
: (4) LONG VERSION        = NO            (5) DISPLAY TO SCREEN = YES     :
: (6) DISPLAY DESTINATION = NONE                                          :
: (7) NO PURGE            = FALSE         (8) NUMBER OF COPIES  = 1       :
:                                                                         :
: **   (E) EXECUTE, (?) HELP, (Q) QUIT TO PREVIOUS MENU, (N) END $ACCESS  ** :
+-------------------------------------------------------------------------+
```

Die Nummer oben rechts auf dem Bild zeigt seinen Platz in der Bild-Hierarchie:
3.7 bedeutet beispielsweise:
3. Stufe (Main-Menu - User Def. - Display Users),
7.Bild innerhalb dieser 3. Hierarchie-Stufe.

Allgemeine Eingabe-Regeln:

Folgende Eingaben gelten für die meisten Bilder:

Q (Quit): Zurück auf das nächsthöhere Bild der Hierarchie.
Auf dem Main-Menu beendet QUI das Programm.

N (End): Auf allen Bildern ausser auf dem Main-Menu: Beendet das Programm.

E (Execute): Auf Bildern mit mehreren Parametern werden alle Parameter des Bildes als gültig aktiviert.
Nummer gemäss Bild (Zahlen in Klammer):
Vor der Eingabe von "E" lassen sich einzelne Parameter über ihre Nummer ansprechen und verändern.
<RET> als Dateneingabe übernimmt den alten Inhalt.

? (Help): Ausgabe englischer Erklärungen zum laufenden Bild.
Ausgabe beenden mit der Eingabe "X".

26.3 ERSTMALIGES GENERIEREN DES SECURITY-SYSTEMS

Wichtige Bemerkung:

Falls $ACCESS auf Ihrem System noch nicht generiert ist, erscheint nach dem Start mit EX $ACCESS die Meldung:

> 0086: SYSTEM ACCESS CONTROL HAS NOT YET BEEN CREATED, PLEASE DO SO BEFORE ATTEMPTING ANY OTHER FUNCTIONS OF THIS UTIILITY, NEWLINE TO CONTINUE
>
> In diesem Fall sollte zuerst die nachfolgend beschriebene Prozedur ausgeführt werden.
>
> **Warnung:**
> Wenn dies nicht getan wird, kann jeder Bildschirm-Benützer an jedem Schirm diese Prozedur ausführen und somit User-, Passwort- und andere Security-Definitionen erstellen, die nur dem Ersteller zugänglich sind !

Prozedur zum erstmaligen Generieren des Security-Systems

Wir empfehlen Ihnen, neben der Lektüre die Prozedur am Bildschirm auszuführen, damit Sie die Bilder vor sich haben.

Diese Kurzbeschreibung ergibt den einfachsten Fall mit 1 Administrator-User (adminuser) und 1 Normal-User (normuser) ohne Passwort-Eingabe.
Für den Ausbau beachten Sie bitte die nachfolgende "Beschreibung der einzelnen Bilder".

Erstmaliges Generieren des Security-Systems (Fortsetzung)

Bild	Eingabe / Bemerkungen
Bild 1.0 (Main Menu):	**MAI** zur SAC-File-Maintenance
Bild 2.0	**CRE** Erstellen eines neuen SAC-Files auf dem System-Disc
Bild 3.12	**E** Ohne Angaben gilt: 276 sektoren für max. 500 User.
Nach der Meldung SYSTEM ACCESS CONTROL CREATED...	
	<RET>
Bild 3.12	**Q**
Bild 2.0	**Q**
Bild 1.0	**POL** Als Policy die
Bild 2.2	**SET** Utility Security auf ADMINISTRATOR setzen.
Bild 3.8	**8** (Utility Security) dann **ADMIN** dann **E**
Nach der Meldung SYSTEM ACCESS CONTROL OPTIONS CREADTED...	**<RET>**
Bild 3.8	**Q**
Bild 2.2	**Q**
Bild 1.0	**USE** je einen User vom Typ
Bild 2.1	**ADD** "Administrator" und "Normal" erstellen:
Bild 3.4	**1** dann **adminuser** dann
	4 (User Type) dann **ADMIN** dann
	E eingeben.
Nach der Meldung THE NEW USER HAS BEEN ADDED...	
	<RET>
Bild 3.4	**1** dann **normuser** dann
	4 (User Type) dann **NORMAL** dann
	E eingeben.
Nach der Meldung THE NEW USER HAS BEEN ADDED...	
	<RET>
dann auf allen Bildern:	**Q** bis $ACCESS beendet ist.

zu lang !

Nun kann an jedem Bildschirm mit einer der folgenden USER-ID's gestartet werden:

- **normuser :** Ohne die Erlaubnis für $ACCESS
- **adminuser:** Mit der Möglichkeit, $ACCESS zu starten.

Es werden **keine Passwörter** verlangt.

26.4 BESCHREIBUNG DER EINZELNEN BILDER (Administrator-User)

Die "System Security" lässt sich durch folgende Massnahmen verfeinern und erhöhen:

- Festlegen einer zeitlichen Sperre, Detach oder Inaktivierung des Users nach einer bestimmten Anzahl falscher User-/Passwort-Eingaben.
- Festlegen, ob überhaupt Passwörter erforderlich sind.
- Festlegen der minimalen Passwortlänge.
- Erweitern der Passwörter durch Zufallszeichen bis auf 8 Stellen durch das System.
- Bestimmen, ob jeder Normal-User seine Passwörter ändern kann.
- Festlegen eines Verfall-Datums für User und/oder Passwörter.

Zur Orientierung beachten Sie bitte die Funktions- und Bild-Hierarchie weiter vorn im diesem Kapitel.

Für alle Bilder gelten überdies die "Allgemeinen Eingaberegeln", die ebenfalls weiter vorn in diesem Kapitel beschrieben sind.

Hinweise:

Dieses Kapitel gilt für **Administrator-User.**

Die Bild-Beschreibung für **Normal-User** finden Sie im Kapitel 26.5

Am Ende jeder Bild-Beschreibung steht unter **"Befehlsmodus"** das Format der Eingaben, wenn $ACCESS im Befehls-Modus abläuft.
Der Befehls-Modus ist im Kapitel 26.6, nach den Bild-Beschreibungen ausführlich beschrieben.

MAIN MENU

SYSTEM ACCESS CONTROL UTILITY (Bild 1.0)

Dieses Bild erscheint stets nach dem Start von $ACCESS.
Es ist das höchstrangige Auswahlbild.

Mögliche Eingaben:

OPT Zum Utility Options Menu: Wahl des Menu- oder Befehls-Modus.

MAI Zum SAC-Maintenance Menu: Aufbau und Reorganisation des SAC-Files.

USE Zum User Definition Menu: Unterhalt von User-Definitionen und Passwörtern.

POL Zum System Policies Menu: Festlegen diverser Parameter über Passwort-Philosophie, Reaktion auf Fehleingaben usw.

QUI oder END
Programm $ACCESS abschliessen.

??? Help: Ausgabe englischer Erklärungen.
Beenden der Ausgabe mit X.

Befehls-Modus:

OPT oder MAI oder USE oder POL oder QUI

UTILITY OPTIONS (Bild 3.0)

Dieses Bild erlaubt die Wahl zwischen Menu- und Befehls-Modus.

Mögliche Eingaben:

1	dann Eingabe: MENU Menu-Modus COMMAND Befehls-Modus. (Beachten Sie den besonderen Abschnitt Ende des Kapitels)
E	Execute: Der angezeigte Modus wird aktiviert.
?	Help: Ausgabe englischer Erklärungen.
Q	Zurück zum Main-Menu.
N	Programm $ACCESS abschliessen.

Befehls-Modus:

MOD=MEN oder MOD=COM

USER DEFINITION (Bild 2.1)

Unterhalt von User-Identifikationen und Passwörtern.

Mögliche Eingaben:

ADD	Einen neuen User definieren
DEL	Einen User löschen.
UPD	Eine User-Definition verändern: User-Typ oder Passwort.
DIS	User-Definitionen ausgeben.
???	Help: Ausgabe englischer Erklärungen.
QUI	Zurück zum Main-Menu.
END	Programm $ACCESS abschliessen.

Befehls-Modus:

OPT=ADD oder OPT=DEL oder OPT=UPD oder OPT=DIS oder OPT=QUI oder OPT=END

ADD USER **(Bild 3.4)**

Definieren eines neuen Users.

Mögliche Eingaben:

1 User Id: Eingabe einer User-Identifikation:
Höchstens 8 Stellen aus Buchstaben, Ziffern oder Bindestrich. Gross- und Kleinbuchstaben werden gleich behandelt.

2 Node Id: Nur für Systeme mit DDP.
LOCALGilt für dieses System (Local node) oder Node-Identifikation eines andern Systems, von dem SUBMIT-Jobs gestartet werden können.

3 Password: Eingabe des Passwortes:
Höchstens 8 Stellen aus ASCII-Zeichen mit einem Hex-Wert von hex 21 - 2B oder 2D - 7E.
Keine Leerstellen und Kommas!
Gross- und Kleinbuchstaben gelten als verschiedene Zeichen.

4 User Type: Eingabe des User-Typs:
ADMINISTRATOR: Kann alle Funktionen von $ACCESS ausführen.
NORMAL: Kann mit $ACCESS nur sein eigenes Passwort mutieren, sonst nichts.

5 User Control String:
Physname und Disc-Einheit des Start-Controlstrings dieses Users.
Dieser Controlstring läuft jedesmal ab, wenn sich dieser User anmeldet.
NONE: Kein Start-Controlstring.

6 Control String Option:
Aktion, falls der User Controlstring abbricht:
ABORT: Bei Controlsting-Fehler oder -Abbruch den Zugriff verweigern.
CONTINUE: Auch bei Controlstring-Fehler oder -Abbruch einen Prozess eröffnen.

7 Expiration Date:
Verfall-Datum des Users.
NONE: Kein Verfall-Datum.
JJ/MM/TT: Datum, nach dem dieser User nicht mehr gültig ist.

ADD USER (Bild 3.4) Fortsetzung

E Die angezeigte User-Definition eröffen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Definition Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

```
USE=userid        PASS=passwort
TYP=ADM oder      TYP=NOR
CSTR=NONE oder    CSTR=c-string(n)
COPT=ABO oder     COPT=CON
EXP=jj/mm/tt oder EXP=NONE
```

DELETE USER (Bild 3.5)

Löschen eines Users.

Mögliche Eingaben:

1 Eingabe einer bestehenden User-Identifikation.

2 Node Id: Nur für Systeme mit ITXNET.
LOCAL: Gilt für dieses System (Local node) oder
Nodeid: Identifikation eines andern Systems.

E Die angezeigte User-Definition löschen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Definition Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

USE=userid
NOD=LOCAL oder NOD=nodeid

UPDATE USER (Bild 3.6)

Aendern von Passwort oder User-Typ eines bestehenden Users.

Mögliche Eingaben:

1 User Id: Eingabe einer User-Identifikation.

2 Node Id: Nur für Systeme mit DDP.
LOCALGilt für dieses System (Local node) oder Node-Identifikation eines andern Systems.

3 Old Password: Eingabe des bisherigen Passwortes.
NONE = kein Passwort erforderlich.
Ist nur zum Aendern des eigenen Passwortes erforderlich.

4 New Password: Eingabe des neuen Passwortes oder NONE=keines.
Die Regeln finden Sie unter dem Bild ADD USER.

5 User Type: Eingabe des User-Typs:
ADM für Administrator,
NOR für Normal,
CURRENT bedeutet: Usertyp nicht ändern

6 User Control String:
Physname und Disc-Einheit des Start-Controlstrings dieses Users.
Dieser Controlstring läuft jedesmal ab, wenn sich dieser User anmeldet.
CURRENT: Der jetzt gültige Controlstring.
physname/gen(n): Name, eventuell Generation und Disc-Unit des Controlstrings.

7 Control String Option:
Aktion, falls der User Controlstring abbricht:
CURRENT: Die jetzt gültige Option.
ABORT: Bei Controlsting-Fehler oder -Abbruch den Zugriff verweigern.
CONTINUE: Auch bei Controlsting-Fehler oder Abbruch einen Prozess eroffnen.

UPDATE USER (Bild 3.6) Fortsetzung

8 Expiration Date:
Verfall-Datum des Users.
CURRENT: Das jetzt gültige Verfall-Datum.
NONE: Kein Verfall-Datum.
JJ/MM/TT: Datum, nach dem dieser User nicht mehr gültig ist.

9 Deactivate Status:
Aktiv- oder Inaktiv-Status des Users.
CURRENT: Der jetzt gültige Status.
ACTIVE: Der User ist aktiv.
DEACTIVE:: Der User ist inaktiviert und kann nicht benützt werden.

E Die angezeigte User-Definition mutieren.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Definition Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

USE=userid NOD=LOCAL oder NOD=node-id
OLD=altpassw oder OLD=NONE
PASS=passwort oder PASS=NONE
TYP=ADM oder TYP=NOR oder TYP=CUR
CSTR=NONE oder CSTR=c-string(n) oder CSTR=CUR
COPT=ABO oder COPT=CON oder COPT=CUR
EXP=jj/mm/tt oder EXP=NONE oder EXP=CUR
STAT=ACT oder STAT=DEA oder STAT=CUR

DISPLAY USERS (Bild 3.7)

User-Definitionen und Passwörter ausgeben.

Mögliche Eingaben:

1 Type of Display: Eingabe einer Ausgabeart:
ALL = alle User oder
SPE = nur einen User gemäss nächstem Feld.
ADM = alle Administrator-Users.

2 User Id: Eingabe des gewünschten Users.

3 Node Id: Nur für Systeme mit DDP.
LOCALGilt für dieses System (Local node) oder
Node-Identifikation eines andern Systems.

4 Long Version:
NO = nur User-Id und User-Typ ausgeben.
YES = zusätzlich Datum der Erstellung, der letzten Modifikation, des letzten Log-On, der letzten Passwortänderung ausgeben.

5 Display to Screen:
YES = Ausgabe auf den Bildschirm oder
NO = Ausgabe auf Drucker oder Disc gemäss nächstem Feld.

6 Display Destination:
NONE = Ausgabe auf den Bildschirm oder Eingabe eines Druckers oder manuellen Spoolfiles, z.B:
(0,LP) oder SP-SAC(2)

7 No Purge (nur bei Ausgabe auf Autospoolfile):
FALSE = Spoolfile nach Druck löschen,
TRUE = Spoolfile nach Druck nicht. löschen.

8 Number of Copies (nur bei Ausgabe auf Autospoolfile):
1 bis 65535.

DISPLAY USERS (Bild 3.7) Fortsetzung

E Die angezeigte User-Definition ausgeben.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Definition Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

DTY=SPE oder DTY=ALL oder DTY=ADM
USE=NONE oder USE=userid
NOD=LOCAL oder NOD=node-id
LON=YES oder LON=NO
SCR=YES oder SCR=NO
DES=ausgabegerät oder DES=NONE
NOP=TRUE oder NOP=FALSE
COP=anzahl

SYSTEM ACCESS CONTROL MAINTENANCE (Bild 2.0)

Aufbau und Reorganisation des SAC-Systemfiles.

Mögliche Eingaben:

CRE	erstmaliges Initialisieren der System Access Control.
REB	Rebuild das SAC-File: Zum Aendern seiner Grösse oder bei korruptem File.
???	Help: Ausgabe englischer Erklärungen.
QUI	Zurück zum Main-Menu.
END	Programm $ACCESS abschliessen.

Befehls-Modus:

OPT=CRE oder OPT=REB oder OPT=QUI oder OPT=END

CREATE SYSTEM ACCESS CONTROL (Bild 3.12)

Erstmaliges Initialisieren der System Access Control.

Mögliche Eingaben:

1 Eingabe der Grössen-Bestimmung:
- DEF (Default) 276 Sektoren, reicht für 500 Users.
- USE Grösse gemäss Anzahl User im Feld 3.
- SEC Grösse in Sektoren gemäss Feld 2.

2 Eingabe der Filegrösse in Sektoren.
- 32 Sektoren = bis 50 Users,
- 68 Sektoren = bis 100 Users,
- 120 Sektoren = bis 200 Users,
- 276 Sektoren = bis 500 Users,
- 584 Sektoren = bis 1000 Users,
- 2460 Sektoren = bis 5000 Users.

3 Eingabe der maximalen Anzahl Users.

E Die angezeigten Angaben ausführen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum SAC Maintenance Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

SIZ=DEF oder SIZ=USE oder SIZ=SEC
NSE=sektoren
NUS=anzahluser

REBUILD SYSTEM ACCESS CONTROL (Bild 3.3)

Aendern der Grösse des SAC-Files oder Rebuild bei beschädigtem SAC-File.

Mögliche Eingaben:

1 System to Rebuild: Disc, wo sich das SAC-File befindet.
CURRENT Das aktuelle SAC-File auf (SYS3).
SPECIFIC Unit wird im Feld 2 angegeben.

2 Disc-Unit, falls Feld 1 = SPECIFIC.
0 bis 255.

3 Scratch Unit:
Eingabe der Disc-Unit für das temporäre Workfile,
0 bis 255.

4 Type of File Size: Eingabe der Grössen-Bestimmung:
CURRENTDie Grösse bleibt unverändert.
USE Grösse gemäss Anzahl User im Feld 6.
SEC Grösse in Sektoren gemäss Feld 5.

5 Eingabe der Filegrösse in Sektoren.
32 Sektoren = bis 50 Users,
68 Sektoren = bis 100 Users,
120 Sektoren = bis 200 Users,
276 Sektoren = bis 500 Users,
584 Sektoren = bis 1000 Users,
2460 Sektoren = bis 5000 Users.

6 Eingabe der maximalen User-Anzahl.

7 Verify Changes:
YES Bereinigen von korrupten Daten oder Lesefehler im SAC-File mit Meldung auf dem Bildschirm. Feld 8 muss YES sein!
NO In diesen Fällen ohne Meldung weiterfahren.

8 Display Changes to Screen:
YES Rebuild-Meldungen auf den Bildschirm ausgeben.
NO Keine Rebuild-Meldungen ausgeben.

REBUILD SYSTEM ACCESS CONTROL (Bild 3.3), Fortsetzung

9 Display Destination:
Ausgabe der Rebuild-Meldungen auf Drucker oder Spoolfile:
NONE Keine Druck-Ausgabe.
Eingabe eines Druckers oder manuellen Spoolfiles, z.B:
(0,LP) oder SP-SAC(2)
Die Druckausgabe enthält auch Meldungen, die am Bildschirm nicht erscheinen.

10 No Purge (nur bei Ausgabe auf Autospoolfile):
FALSE = Spoolfile nach Druck löschen
TRUE = Spoolfile nach Druck nicht. löschen.

11 Number of Copies (nur bei Ausgabe auf Autospoolfile):
1 bis 65535.

E Die angezeigte File-Rebuild-Prozedur ausführen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum SAC Maintenance Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

TYP=CURRENT oder TYP=SPECIFIC
UNI=discunit
SUN=scratchunit
DES=ausggerät oder DES=NONE
NOP=TRUE oder NOP=FALSE
COP=anzahl
SIZ=DEF oder SIZ=USE oder SIZ=SEC
NSE=sektoren
NUS=anzahluser

SYSTEM POLICIES (Bild 2.2)

Festlegen diverser Parameter für das System Access Control-System:
Z.B: Ob Passwort erforderlich, Passwortlänge, Verhalten bei falscher User-Id/Passwort-Eingabe usw.

Mögliche Eingaben:

SET	Festlegen oder ändern der Parameter.
DIS	Ausgabe der aktuellen Parameter
???	Help: Ausgabe englischer Erklärungen.
QUI	Zurück zum Main-Menu.
END	Programm $ACCESS abschliessen.

Befehls-Modus:

OPT=SET oder OPT=DIS oder OPT=QUI oder OPT=END

SET POLICIES (Bild 3.8)

Festlegen und ändern von Parametern.

Mögliche Eingaben:

1 Password required: Passwort erforderlich?
YES Zu jeder User-Id muss ein Passwort eingegeben werden.
NO Es bestehen keine Passwörter.

2 Password Minimum Length:
0 bis 8 = Minimale Passwortlänge. Länge 0 ermöglicht auch <RET> als Passwort.

3 Password Change Intervall:
0 bis 65535 = Tage, nach denen die Passwörter spätestens gewechselt werden müssen. Nach Ablauf dieser Zeit erscheint beim SYSTEM READY FOR LOG-ON eine Meldung.

4 Short Password Option:
PAD (Default): Passwörter von weniger als 8 Stellen werden mit Leerstellen auf 8 Stellen ergänzt.
EXTEND: Passwörter von weniger als 8 Stellen werden vom System mit mit Zufallszeichen auf 8 Stellen ergänzt. Beim Log-on sind alle 8 Stellen einzugeben.
Z.B: Das definierte Passwort ABC
wird beispielsweise zu ABCgR8!x

5 Logon Retry Limit: Anzahl Fehl-Eingaben beim Log-On, bis das System gemäss den folgenden Feldern reagiert:
0 bis 255.

6 Block Terminal Option: Reaktion beim Erreichen der Limite für Fehleingaben:
NONE Nach dem Druck der BREAK-Taste kann wieder versucht werden.
DETACH Der Bildschirm wird detached.
IGNORE Der Bildschirm wird für die Zeitdauer gemäss dem folgenden Feld blockiert.

7 Logon Lockout Time: Zeit in Sekunden, bis nach der Retry Limit wieder Eingaben möglich sind:
0 bis 65535 Sekunden.

SET POLICIES **(Bild 3.8), Fortsetzung**

8 Utility Security: Sicherheit des Utilities $ACCESS:
- NONE — Jeder User kann alle $ACCESS-Funktionen ausführen.
- PARTIAL — Jeder User kann $ACCESS starten, jedoch: Normal-User können nur ihre eigenen Passwörter sehen/ändern. Administrator-User können alle Funktonen ausführen.
- ADMINISTRATOR — Nur Administrator-User können $ACCESS starten.

9 Node Id at Logon: Nur für Systeme mit ITXNET.
- YES — Die Node-Identifikation wird beim Log-On verlangt.
- NO — Die Node-Identifikation wird beim Log-On nicht verlangt.

10 Force Password Policy.
Verhalten des Systems, wenn ein Passwort abgelaufen ist:
- YES: Der User muss ein neues Passwort festlegen.
  Er erhält die Aufforderungen:
  PLEASE CHANGE YOUR PASSWORD
  S)ET PASSWORD OR L)OGOFF
  Wählt er S, muss er das neue Passwort zweimal eintippen.
  Das neue Passwort darf nicht dasselbe sein wie bisher.
  ---> Vorsicht:
  Das Passwort ist nirgends dokumentiert!
- NO: Der User wird nur informiert, dass sein Passwort nicht mehr gültig ist. Der Administrator legt das neue Passwort fest.

11 Deactivate User Policy.
- YES: Wenn ein User die als RETRY LIMIT festgelegte Anzahl Versuche der Passwort-Eingabe überschreitet, wird er inaktiviert und muss vom Administrator reaktiviert werden: Unter Update User, DEACTIVATE STATUS auf ACTIVE setzen.
- NO: Der User wird nicht inaktiviert, wenn er die Retry Limit erreicht.

SET POLICIES **(Bild 3.8), Fortsetzung**

12 Access Utility Log Option.
YES: Alle mit $ACCESS ausgeführten Modifikationen werden ins Audit-Trail-File geschrieben (falls generiert).
NO: Keine Eintragungen ins Audit-Trail-File.

13 Remote Logon Policy.
PROMPT: Für Remote Logon gelten die User-Angaben vom lokalen System.
TRANSPARENT: Für Remote Logon gelten die User-Angaben vom System, an welches das Terminal mit NELOGON angeschlossen wird.
EXCEPTION: Für Remote Logon gilt TRANSPARENT; falls dies misslingt, jedoch: PROMPT.

14 Sysgen Security.
ADMIN: Nur Admin-User können $SYSGEN starten.
PARTIAL: Normal-User können $SYSGEN nur zum Kontrollieren starten, jedoch nichts verändern. Admin-User können auch Aenderungen ausführen.
NONE: Keine Einschränkung für $SYSGEN.

E Die angezeigten Parameter ausführen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum System Policies Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

LPAS=YES oder LPAS=NO LEN=passwortlänge
CHA=tage SHO=PD oder SHO=EXTEND
RET=anzahl-versuche oder RET=UNL
BLO=NONE oder BLO=DET oder BLO=IGN
LOC=lockout-zeit
SEC=NONE oder SEC=PARTIAL oder SEC=ADM
NOD=YES oder NOD=NO FPAS=YES oder FPAS=NO
DEA=YES oder DEA=NO LOG=YES oder LOG=NO
RLOG=PROMPT oder TRANSPARENT oder EXCEPTION
SYSG=NONE oder PARTIAL oder ADMINISTRATOR

DISPLAY POLICIES (Bild 3.9)

Ausgabe der Policy-Parameter.

Mögliche Eingaben:

1 Display to Screen:
YES Ausgabe auf den Bildschirm.
NO Ausgabe auf Drucker oder Spoolfile.

2 Display Destination:
NONE Ausgabe nur auf den Bildschirm.
Eingabe eines Druckers oder manuellen Spoolfiles, z.B:
(0,LP) oder SP-SAC(2)

3 No Purge (nur bei Ausgabe auf Autospoolfile):
FALSE = Spoolfile nach Druck löschen,
TRUE = Spoolfile nach Druck nicht. löschen.

4 Number of Copies (nur bei Autospoolfile):
1 bis 65535.

E Die angezeigte Ausgabe ausführen.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum System Policies Menu.

N Programm $ACCESS abschliessen.

Befehls-Modus:

SCR=YES oder SCR=NO
DES=ausggerät oder DES=NONE
NOP=TRUE oder NOP=FALSE
COP=anzahl

26.5 BESCHREIBUNG DER BILDER FUER NORMAL-USER

MAIN MENU (Bild 1.1)

Dieses Bild erscheint stets nach dem Start von $ACCESS. Es ist das höchstrangige Auswahlbild.

Mögliche Eingaben:

OPT Zum Utility Options Menu: Wahl des Menu- oder Befehls-Modus.

USE Zum User-Attribute Menu: Passwort ändern und User-Attribute kontrollieren.

QUI Programm $ACCESS abschliessen.

??? Help: Ausgabe englischer Erklärungen.
Beenden der Ausgabe mit X.

UTILITY OPTIONS MENU (Bild 3.0)

Dieses Bild erlaubt die Wahl zwischen Menu- und Befehls-Modus.

Mögliche Eingaben:

1 dann Eingabe:

MENU	Menu-Modus
COMMAND	Befehls-Modus. (Beachten Sie den besonderen Abschnitt Ende des Kapitels)

E Execute: Der angezeigte Modus wird aktiviert.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum Main-Menu.

N Programm $ACCESS abschliessen.

USER ATTRIBUES (Bild 2.3)

Unterhalt von User-Identifikationen und Passwörtern.

Mögliche Eingaben:

UPD zu Update Password:
Aendern des eigenen Passwortes

DIS User-Attribute und Passwort ausgeben.

??? Help: Ausgabe englischer Erklärungen.

QUI Zurück zum Main-Menu.

END Programm $ACCESS abschliessen.

UPDATE PASSWORD (Bild 3.10)

Aendern des eigenen Passwortes.

Mögliche Eingaben:

1 Old Password: Eingabe des bisherigen Passwortes.
NONE = kein Passwort erforderlich.

2 New Password: Eingabe des neuen Passwortes oder
NONE=keines.
Die Regeln finden Sie unter dem Bild ADD USER.

E Das angezeigte Passwort mutieren.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Attribute Menu.

N Programm $ACCESS abschliessen.

DISPLAY ATTRIBUTES (Bild 3.11)

User-Definitionen und Passwörter ausgeben.

Mögliche Eingaben:

1 Display to Screen:
YES = Ausgabe auf den Bildschirm oder
NO = Ausgabe auf Drucker oder Disc gemäss nächstem Feld.

2 Display Destination:
NONE = Ausgabe auf den Bildschirm oder
Eingabe eines Druckers oder manuellen Spoolfiles, z.B:
(0,LP) oder SP-SAC(2)

3 No Purge (nur bei Ausgabe auf Autospoolfile):
FALSE = Spoolfile nach Druck löschen,
TRUE = Spoolfile nach Druck nicht. löschen.

4 Number of Copies (nur bei auf Autospoolfile):
1 bis 65535.

E Die angezeigte User-Definition ausgeben.

? Help: Ausgabe englischer Erklärungen.

Q Zurück zum User Attribute Menu.

N Programm $ACCESS abschliessen.

26.6 BEFEHLS-MODUS FUER $ACCESS

Der Befehls-Modus lässt sich in folgenden Formen anwenden:
- Ausführung von $ACCESS ab einem Controlstring, ohne Eingriff vom Bildschirm.
- Befehlseingabe vom Bildschirm. Vor und nachher ist der Menu-Modus möglich.

Allgemeine Regeln für die Befehls-Eingabe:

Alle Eingaben, die im Menu-Modus am Bildschirm gemacht werden, müssen als je ein Befehl eingegeben werden bzw. im Controlstring enthalten sein.
Das Befehls-Format finden Sie jeweils am Ende jeder Bildbeschreibung.
Auch die Execute-Eingabe (E) für ein Bild, die Eingaben für die Rückkehr zu einem andern Bild und die Eingabe für Programmende sind als Befehle einzugeben!

Empfehlung:
Vor dem Aufbauen der Befehlsfolge dieselbe Aktion im Menu-Modus durchführen und alle Eingben genau notieren.

Ausführung ab Controlstring:

Start:
EX $ACCESS **CMD** mit dem Parameter **CMD**!

Befehle:
Jeder Befehl und jede Eingabe je eine Zeile.

Beispiel:

Eröffnen eines neuen Users:

```
EX $ACCESS CMD
USE                 Wahl von "User Definition"
OPT=ADD             Wahl von "Add User"
USE=HANS23          Der User heisst "HANS23"
TYP=NORMAL          Es ist ein Normal-User
PASS=XYZ-*          Sein Passwort ist "XYZ-*
E                   Execute der User-Angaben
N                   Ende von $ACCESS
```

Ausführung des Befehls-Modus am Bildschirm:

Diese Variante eignet sich für Personen, die sehr häufig mit $ACCESS arbeiten und die Menubilder nicht mehr als Hilfe brauchen.

Nach dem Start mit $ACCESS (Ohne CMD) wird auf dem Option-Menu der Modus COMMAND gewählt. Anschliessend läuft das Programm im Befehls-Modus.

Beispiel:

Eröffnen eines neuen Users:
Eingaben sind fett, Ausgaben normal gedruckt.

```
EX $ACCESS
Bild 1.0:
 OPT                          Wahl von "Options"
Bild 3.0:
 1 dann COMMAND dann E        Wahl des Command-Modus.
Utility Options               Das Programm meldet nun
                              nach jedem Befehl, wo
                              es steht.
----> Von hier weg nur noch Befehls-Eingaben:
Q                             Zurück zum Main Menu
System Access Control Utility
USE                           Wahl von "User Definitions"
User Definition
OPT=ADD                       Wahl von "Add User"
Add User
USE=HANS23                    Der User heisst "HANS23"
TYP=NORMAL                    Es ist ein Normal-User
PASS=XYZ-*                    Sein Passwort ist "XYZ-*
E                             Execute der User-Angaben
The new User has been added successfully
N                             Ende von $ACCESS
```

27. KONVERSION VON OBJEKTPROGRAMMEN ($OBJCONV)

Funktionen:

Umwandeln von Objekt-Programmen, die für IMOS III, IMOS V oder IMX erstellt wurden, in IRX/ITX-Programme.
Die Konversion ist mit Kopieren verbunden.

Aendern der Auflösung der COBOL-ACCEPT-Befehle:
- Mit EXACT = Bei Eingabe der vollen Feldlänge läuft das Programm weiter
  oder:
- Ohne EXACT = Jede Feld-Eingabe muss mit NEWLINE abgeschlossen werden, auch nach Eingabe der vollen Feldlänge.

Durchführung:

```
[ AS BI physname-1 (n) OW ]      nur für Konversion
                                 von IMOS/IMX-Programmen

  AS BO physname-2 (n) / NE s AP \
                       \   OW    /

  EX $OBJCONV [ EXACT=x ]
```

physname-1 = Name des bestehenden IMOS/IMX-Objektprogramms.

physname-2 = Name des IRX/ITX-Objektprogramms

s = Grösse des IRX/ITX-Objektprogramms in Sektoren. Es wird 1 bis 2 Sektoren grösser als das IMOS-Objektprogramm.

EXACT=x = Angabe, ob ein IMOS/IMX- in ein ITX-Objectprogramm zu konvertieren ist und
Wahl, wie die COBOL-ACCEPT-Befehle aufzulösen sind, wobei x = :

0 oder keine EXACT=x-Angabe:
BI ist ein IMOS/IMX-Programm,
BO ein ITX-Programm
(Konversion)
ACCEPT's wie ohne EXACT.

1 BI ist ein IMOS/IMX-Programm,
BO ein ITX-Programm
(Konversion)
ACCEPT's wie mit EXACT
(IMOS-artig).

Für EXACT=2 oder 3 ist das BO-Programm mit OWN zuzuteilen:

2 BO ist ein IRX/ITX-Programm.
ACCEPT's wie mit EXACT
(IMOS-artig).

3 BO ist ein IRX/ITX-Programm.
ACCEPT's wie ohne EXACT.

Beispiele:

a) Ein IMOS/IMX-Programm in ein ITX-Programm konvertieren. Alle ACCEPS's sollen "IMOS-artig", d.h wie mit EXACT ablaufen.

```
AS BI P-IMOS (1) OW
AS BO P-ITX  (1) NE 100 AP
EX $OBJCONV EXACT=1
```

b) Im ITX-Programm P-DEBI20 alle ACCEPS's so umsetzen, dass sie "IMOS-artig", d.h wie mit EXACT ablaufen.

```
AS BO P-DEBI20 (1) OW
EX $OBJCONV EXACT=2
```

28. SYSTEM-DISC REKONSTRUIEREN

VARIANTEN:

a) Boot-Tape
Sind alle Teile des Systemdisc (SYS1, SYS2 und SYS3) auf derselben Disc-Einheit, kann der Systemdisc als "Boot-Tape" gesichert werden. Ein Boot-Tape kann sich auch über mehrere Streamer-Bänder erstrecken. (Vgl. Kapitel "$STREAM", Befehl BUILD).

Falls die Systemdisc-Teile SYS1, SYS2 und SYS3 nicht auf derselben Disc-Einheit liegen, muss das System auch als mehr-teilige Kopie gesichert werden, wahlweise auf Streamer- oder Magnet-Band:

b) Streamer-Sicherung, jedoch nicht als Boot-Tape
Jede Disc-Einheit, die System-Elemente (SYS1, SYS2 oder SYS3) enthält, wird auf ein eigenes Streamer-Band gesichert.
(Vgl. Kapitel "$STREAM", Befehl BACKUP).

Für das Rekonstruieren des System-Disc ist in jedem Fall ein **"Boot-Tape** für den entsprechenden System-Typ und ITX-Release erforderlich.

DURCHFUEHRUNG:

1. Das System muss ausgeschaltet sein (Power off):

   Das Streamerband **"Boot Tape"** einlegen.

2. Das **System starten** (einschalten)
   Es werden verlangt: Date und Time,
   Disc address (in der Regel 1000)

3. Die Disc-Einheit für den **gewünschten Systemdisc initialisieren:**
   Auf dem Auswahlbild "Tape Boot function to perform":
   IN wählen
   Es folgt derselbe Dialog wie nach EX $DINT.
   Wenn der System-Disc noch einwandfrei initialisiert ist, genügt ein "Volume Software Rebuild, Frage D209 = 2.

4. Das Boot-System auf die Original-Systemdisc-Einheit kopieren:
   Auf dem Auswahlbild "Tape Boot function to perform":
   MO wählen (Move Boot System to Disc)
   Es werden verlangt:
   VOLSERNO = **Packnummer** des Systemdisc
   IF ITX SOFTWARE.. = **RE** (Replace with Boot)

5. Das Boot-System starten:
   Auf dem Auswahlbild "Tape Boot function to perform":
   RE wählen (Reboot from Disc)
   Es folgt der normale System-Start-Dialog.

Ist das Boot-Tape gleich System-Kopie (Variante a), ist damit die Rekonstruktion des Systemdiscs abgeschlossen.

---> Fortsetzung für Variante b), Sicherung auf Streamer-Band, jedoch nicht als Boot-Tape, siehe nächste Seite!

Fortsetzung für **Variante b),**
System-Sicherung auf Streamer-Band, jedoch nicht als Boot-Tape:

7. Das Streamerband "Boot Tape" entfernen

8 Das Streamerband **"Original-System-Kopie"** einlegen.

9. Das Original-System ab Streamer auf den System-Disc kopieren:
   EX $STREAM
   Für jedes Streamer-Band je 1 Befehl:
   RES kopiename (n MT) (n) SO

   Gewisse Systemfiles sind schon auf dem Disc vorhanden und erzeugen die Meldung
   FILE NOT RESTORED...

   $STREAM mit **QU** abschliessen.

10. Das Steamer-Band entfernen

11. Das **System normal starten.**
    Power off und wieder einschalten
    oder:
    Schnell-Variante ohne auszuschalten:
    Befehl ESCAPE,
    dann Control-B,
    System Disc address: In der Regel 1000

Damit ist die Systemdisc-Rekonstruktion abgeschlossen.

- Notizen -

29. DER TEXT-EDITOR ($EDIT)

29.1 MOEGLICHKEITEN

Der Text-Editor ermöglicht das Erstellen, Aendern und Kopieren von Source-Programmen, Control-Strings und Datendateien mit einer Recordlänge von maximal 80 Bytes (ohne VLI), fix oder variabel.
Control-Strings und COBOL Source-Programme weisen in der Regel variable Records auf.

Als Eingabe-Medien können Bildschirm, Disc oder Magnetband verwendet werden.
Disc-Dateien können auch relativ oder indexed organisiert sein.

Als Ausgabe-Medien sind Bildschirm, Disc, Magnetband oder Drucker möglich.
Alle Ausgabe-Dateien werden sequentiell organisiert.

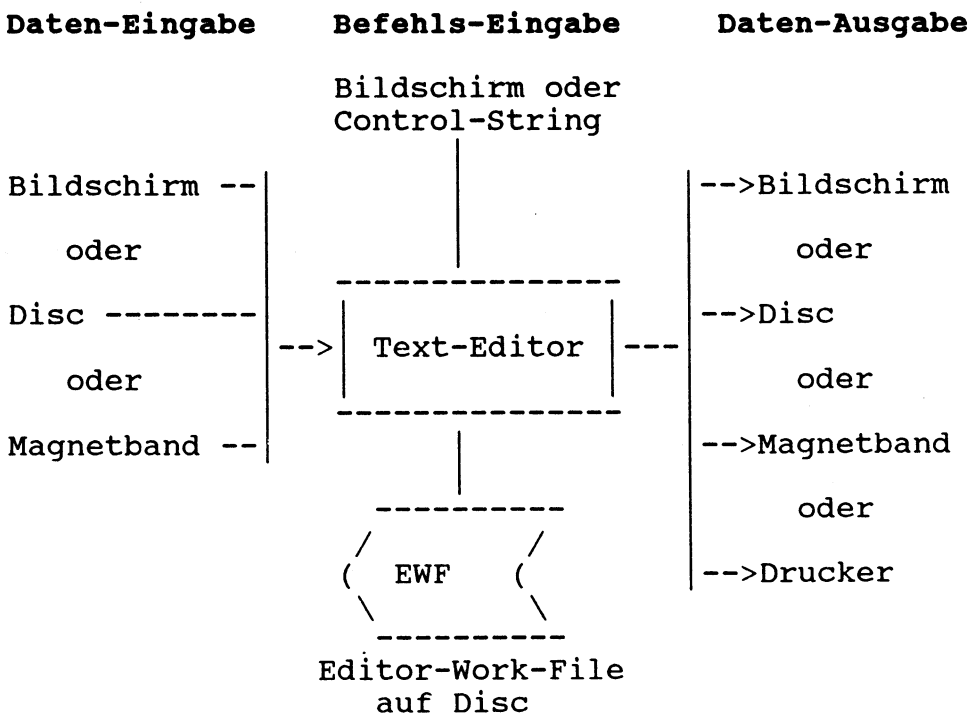
Der Text-Editor benötigt als Arbeitsdatei ein Text-Editor-Work-File **(EWF)** auf Disc.

Die Programm-Steuerung erfolgt durch verschiedene Text-Editor-Befehle.
Diese werden meistens auf dem Bildschirm eingegeben, können aber auch im Rahmen eines Control-String ab einer sequentiellen Datei eingelesen werden.

Hinweis:

Das ITX-Betriebssystem enthält neben dem Texteditor $EDIT den SCL-Editor, welcher mit dem Systembefehl EDIT gestartet wird.
Die Beschreibung jenes Editors finden Sie im Kapitel "SCL-Editor".

Schematische Darstellung



29.2 DAS EDITOR WORK FILE (EWF)

29.2.1 FUNKTION

Das EWF ist die Arbeitsdatei des Text-Editors. Bevor eine bestehende Datei verarbeitet werden kann, muss sie ins EWF eingelesen werden.
Sämtliche Veränderungen, Einfügungen und Löschungen von Records sowie alle Record-Suchfunktionen werden im EWF durchgeführt.
Während eines Text-Editor-Laufes kann der aktuelle Inhalt des EWF jederzeit auf eine Ausgabe-Datei geschrieben werden.

29.2.2 ERSTELLUNG UND ZUORDNUNG

Auf einem neu generierten System muss jedes EWF erstmals mit folgendem ASSIGN-Befehl erstellt und zugleich dem Text-Editor zugeordnet werden:

AS EWF EWFxx (n) **PR NE** 500

xx = In der Regel eine Bildschirm-Nummer, da für jeden Schirm ein eigenes EWF verwendet wird. Für besondere Verarbeitungen können aber auch Editor Work Files mit anderen Namen definiert werden.

n = Disc-Einneit.

PR = Obligatorisch, da das EWF nur immer einer Verarbeitung gleichzeitig zugeordnet werden kann.

500 = Standardgrösse. Für besondere Verarbeitungen sind auch andere Grössen möglich.

Ist ein EWF einmal definiert, kann es für alle weiteren Text-Editor-Läufe mit folgendem ASSIGN-Befehl zugeordnet werden:

AS EWF EWFxx (n)

29.2.3 RECORDSTRUKTUR DES EWF

Der Text-Editor verarbeitet Records von fixer oder variabler Länge mit einer Maximallänge von 80 Bytes (zuzüglich VLI bei variabler Länge).

Relative Position

Jedem Record im EWF wird eine relative Position zugeordnet. Sie dient in verschiedenen Text-Editor-Befehlen als Suchbegriff. Im Bildschirm-Dialog mit dem Text-Editor wird sie als Nummer des laufenden Records nach der Ausführung jedes Befehls ausgegeben.
Die relative Position ist kein Record-Bestandteil; sie wird vom Text-Editor laufend errechnet und im Memory gespeichert.

Seiten/Zeilen-Nummern

In Dateien, wo die ersten 1 bis 8 Stellen jedes Records eine Nummer darstellen, kann diese als Seiten/Zeilennummer (S/Z-Nummer)angesprochen werden.

Dieses rein numerische Feld lässt sich in verschiedenen Text-Editor-Befehlen als Suchbegriff verwenden.

In COBOL-Source-Programmen ist die S/Z-Nummer 6-stellig.

Control-String-Records enthalten keine Seiten/Zeilen-Nummern.

In Datenfiles, wo die ersten 1 bis 8 Stellen jedes Records eine (aufsteigende) Nummer darstellen, lässt sich diese als S/Z-Nummer ansprechen.

Innerhalb einer Datei sollte die S/Z-Nummer für alle Records gleich lang gewählt werden.

29.2.4 RECORD-EINFUEGUNGEN UND LOESCHUNGEN

Die relativen Positionen werden laufend nachgeführt, so dass die Records im EWF stets lückenlos aufsteigend numeriert sind.

Die Seiten/Zeilen-Nummern bleiben durch solche Manipulationen unverändert, eine Neunumerierung findet nicht statt.

29.2.5 SUCHFUNKTIONEN NACH SEITEN/ZEILEN-NUMMERN

Durch den Text-Editor verarbeitete Dateien können in den ersten 1 bis 8 Zeichen jedes Records eine S/Z-Nummer enthalten.
Text-Editor-Funktionen, welche eine Suchoperation nach S/Z-Nummer beinhalten, führen den Suchvorgang wie folgt aus:

- Ist die gesuchte S/Z-Nummer grösser als die laufende, werden nur die auf den laufenden Record folgenden Records sequentiell untersucht.

- Ist die gesuchte Seiten/Zeilen-Nummer kleiner als die laufende, werden nur die dem laufenden Record vorangehenden Records sequentiell rückwärts untersucht.

Es wird empfohlen, Suchfunktionen nach S/Z-Nummer nur in Editor-Work-Files zu verwenden, in denen die S/Z-Nummer-Positionen aller Records numerisch sind und die eine aufsteigende Sortierung nach S/Z-Nummer aufweisen.
In allen Records eines Files sollten gleich viele Stellen für die S/Z-Nummer vorgesehen sein.

29.3 STARTEN DES TEXT-EDITORS

29.3.1 DATEI-ZUORDNUNGEN

Das EWF muss jedem Text-Editor-Lauf zugeteilt werden.

Für jede Ein-und/oder Ausgabe-Datei auf Disc, Kassette oder Lochkarten muss ein ASSIGN-Befehl eingegeben werden.

Ist eine Druckausgabe vorgesehen, muss ein entsprechender ASSIGN-Befehl eingegeben werden. Wird mit dem Text-Editor-Befehl PRINT gedruckt, lautet die Drucker-Zuordnung wie folgt:

AS LO (n, / **LP** \ ) Direkter Druck oder
\ **DLP**/ Auto-Spooling

oder:

AS LO physname (n) **NE s** AP **SP** NO

Erstellen eines manuellen Spool-Files

Eine Disc- oder Kassetten-Eingabedatei kann bei Bedarf durch die Ausgabe überschrieben werden. Dazu wird nur ein ASSIGN-Befehl benötigt.
Handelt es sich um eine Disc-Datei, muss die zugeteilte Anzahl Sektoren für die Aufnahme der neuen Version ausreichen und die Datei als OW zugeordnet werden bzw. als PR definiert sein.

29.3.2 PROGRAMM-START

Das Text-Editor-Programm wird mit dem folgendem EXECUTE-Befehl gestartet:

EX $EDIT [,**RE**COVER]

RECOVER = Der bereits bestehende Inhalt des EWF bleibt erhalten.

Ein Start ohne RECOVER-Angabe löscht den Inhalt des EWF und stellt es als leere Arbeitsdatei zur Verfügung.

Nach einem erfolgreichen Start mit RECOVER wird die folgende Meldung auf den Bildschirm ausgegeben:

```
RECOVERY SUCCESSFUL OF FILE BUILT: DATE = xx/xx/xx
                                   TIME = yy:yy:yy

xxxxxxxxxxxxxx
1 C?
```

DATE und TIME = Datum und Abschluss-zeit des letzten Text-Editorlaufs mit diesem EWF.

xxxxxxxxx = Inhalt des ersten Records

1 C? = Das EWF ist auf diesem ersten Record positioniert.

29.4 UEBERSICHT UBER DIE TEXT-EDITOR-BEFEHLE

Befehls-Eingabe

Alle Befehle können mit Gross- oder Kleinbuchstaben eingegeben werden.

Am Bildschirm

Die Text-Editor-Funktionen werden meistens vom Bildschirm aus gesteuert. Nach dem Start des Programms (mit EX $EDIT) verlangt der Text-Editor mit der folgenden Meldung jeweils einen Befehl:

n C?

n = relative Position des laufenden Records im EWF

In Control-Strings

Für Text-Editor-Läufe im Rahmen von Control-Strings müssen alle Datei-Zuordnungen, der EX $EDIT-Befehl und sämtliche Text-Editor-Befehle mit den dazugehörigen Eingaben im Control-String enthalten sein.
Sind im Control-String alle Text-Editor-Befehle bis QUIT enthalten, wird der String nach dem Texteditor fortgesetzt.

Enthält der Control-String nur Datei-Zuordnungen, EX $EDIT und keine oder nur einen Teil der Text-Editor-Befehle, wird am Control-String-Ende der Text-Editor-Dialog am Bildschirm fortgesetzt.
Beim Abschluss des Text-Editors am Bildschirm mit dem Befehl QUIT wird der Control-String beendet.
Wurde der Control-String, welcher EX $EDIT enthält, von einem andern Control-String mit EX .... LINK aufgerufen, erfolgt bei Abschluss des Text-Editors der Rücksprung in den aufrufenden Control-String.

STEUER-BEFEHLE

ASSIGN	Festlegen verschiedener Parameter, die für andere Text-Editor-Befehle gelten.
DCL	Festlegen der Anzahl Stellen, welche die S/Z-Nummer darstellen.
ERASE	Löschen des Bildschirms.
TAB	Setzen oder Löschen von Tabulations-Positionen im Record.
GRID	Ausgabe eines Spaltenrasters
AGAIN	Wiederholen des letzten Editorbefehls

LESEN UND SCHREIBEN VON DATEIEN

CONCAT	Lesen einer Eingabedatei oder eines Teils derselben ins EWF.
PRINT	Ausdrucken eines oder mehrerer Records aus dem EWF.
SAVE	Schreiben einer Ausgabedatei aus dem EWF.

SUCHEN VON RECORDS IM EWF

Mit Ausgabe der gefundenen Records auf dem Bildschirm.

DISPLAY	Ausgabe von EWF-Records aufgrund der S/Z-Nummer oder der Positionsnummer.
FIND	Suchen eines bestimmten EWF-Records aufgrund der S/Z-Nummer oder der Positionsnummer.
POSITION	Positionieren des EWF um eine bestimmte Anzahl Record-Positionen vor- oder rückwärts.
SEARCH	Suchen eines EWF-Records aufgrund einer Record-Inhaltsangabe.

VERAENDERN, EINFUEGEN ODER LOESCHEN VON RECORDS IM EWF

CHANGE Verändern eines EWF-Records.

DELETE Löschen von EWF-Records.

INSERT Einfügen neuer Record ins EWF.

MOVE Kopieren von EWF-Records an eine andere Position im EWF.

RESEQUENCE Einsetzen neuer Seiten/Zeilen-Nummern in eine Folge von EWF-Records.

STRING Ersetzen von Zeichenfolgen im EWF durch andere Zeichen.

ADJUST Verschieben aller Daten in einem Record nach links oder rechts.

ABBRUCH UND ABSCHLUSS DES TEXT-EDITOR-LAUFES

QUIT Abschluss des Text-Editors.

Wichtig:

Bei einem Abschluss mit QUIT bleibt der Inhalt des EWF erhalten, nicht aber, falls der Text-Editor abgebrochen wird mit der Break-Sequenz und "A".

29.5 BEISPIELE VON TEXT-EDITOR-LAEUFEN

Einschliesslich System-Befehle für den Start.
Die einzelnen Befehls-Parameter werden in den Befehls-Beschreibungen erklärt.

Aufbau einer neuen Datei

AS EWF EWFxx (n)	EWF zuordnen
AS B physname (n) NE, sekt	Datei vorbereiten (mit NE-Parameter)
EX $EDIT	Start des Text-Editors
INSERT	Eingabe neuer Records
xxxxxxxxx	1 Zeile = 1 Record
yyyyyyyyy	
(NL)	Ende der Eingabe
SAVE B	Schreiben der Datei
QUIT	Abschluss

Kopieren (auch eines Teils) einer Kassetten-Datei auf Disc

AS EWF EWFxx (n)	EWF zuordnen
AS A physname (CA)	Kassetten-Datei zuordnen
AS B physname (n) NE, sekt	Disc-Datei vorbereiten (mit NE)
EX $EDIT	Start des Text-Editors
CONCAT A [,*n,m]	Kasetten-Datei ins EWF einlesen [n.bis m.Record]
SAVE B	Das EWF auf die Disc-Datei Datei übertragen
QUIT	Abschluss

Verändern einer bestimmten Datei
mit Zurückschreiben an den alten Platz

AS EWF EWFxx (n)	EWF zuordnen
AS A physname (n) OW	Datei zuordnen (als OWN)
EX $EDIT	Start des Text-Editors
CONCAT A	Datei ins EWF einlesen
CHANGE *n	Einen Record verändern:
xxxxxxxxx	neuer Inhalt
FIND *n **DELETE**	Einen Record suchen und löschen
FIND *n	Einen Record suchen und
INSERT *	dahinter einen neuen Record einfügen:
xxxxxxxx	neuer Record
(NL)	Ende der Eingabe
SAVE A	Zurückschreiben des EWF auf die Datei
QUIT	Abschluss

Ausdrucken einer magnetischen Datei

AS EWF EWFxx (n)	EWF zuordnen
AS A physname (n), RE	Datei zuordnen
AS LO (LP)	Drucker zuordnen
EX $EDIT	Start des Text-Editors
CONCAT A	Datei ins EWF einlesen
PRINT ALL	Ausdrucken ganzes EWF.
QUIT	Abschluss

Aus (Teilen von) bestehenden Dateien eine neue Datei aufbauen

mit Kontrolle der neuen Datei

AS EWF EWFxx (n)	EWF zuordnen
AS A physname (n)	bestehende Dateien
AS B physname (n)	zuordnen
AS C physname (n) NE sekt	Neue Datei vorbereiten (mit NE-Parameter)
EX $EDIT	Start des Text-Editors
CONCAT A [,*n,m]	Einlesen [eines Teils] der Datei A
CONCAT B [,*n,m]	Einlesen [eines Teils] der Datei B
DISPLAY *n,m	Ausgabe des EWF-Inhalts (Kontrolle)
SAVE C	Schreiben der neuen Datei
QUIT	Abschluss

29.6 TEXT-EDITOR-BEFEHLE (ALPHABETISCHE FOLGE)

EINGABE-FORMAT

Alle Text-Editor-Befehle beginnen mit einem Befehlscode (z.B. FIND, INSERT, SAVE usw.), von dem nur die ersten zwei Zeichen erforderlich sind. Vor dem Befehlscode sind keine Leerstellen oder andere Zeichen erlaubt.
Weist ein Befehl Parameter auf, muss nach dem Befehlscode eine Leerstelle, aber kein Komma folgen.

Eine Folge von Parametern wird teils durch Komma, teils durch Klammern und teils durch Bindestriche getrennt. Nach jedem Komma sowie ausserhalb von Klammern sind beliebig viele Leerstellen erlaubt.

Auf dem Bildschirm zeigt die folgende Meldung an, dass ein Befehl eingetippt werden kann:

```
n C?     n = Position des laufenden
             Records im EWF
```

Drückt man anstelle eines Befehls nur die NEWLINE-Taste, erscheint der im EWF folgende Record auf dem Bildschirm und die Nummer des laufenden Records wird um 1 erhöht.

Steht an der ersten Stelle eines "Befehls" ein Stern (*), wird die Eingabe als Kommentar betrachtet und die Aufforderung n C? nochmals ausgegeben.

Pro Zeile können einer oder mehrere Befehle eingegeben werden:
Steht nur 1 Befehl auf einer Zeile, wird er mit NEWLINE abgeschlossen.
Mehrere Befehle auf derselben Zeile müssen mit einem Punkt getrennt werden. Nach dem letzten Befehl wird am Bildschirm NEWLINE gedrückt.

Abkürzungen in den Befehls-Beschreibungen

(NL) = Hier ist die NEWLINE-Taste zu drücken
(nur in den Beispielen angegeben)

ssssss
ttttttt = eine Seiten/Zeilen-Nummer, je nach
nnnnnn DCL-Befehl 1- bis 8-stellig.

n = Eine relative Position oder eine
m Record-Anzahl (1-65535)

Die bei der Beschreibung der System-Befehle verwendeten Befehlsformat-Regeln gelten auch für dieses Kapitel.

In Text-Editor-Befehlen sind alle im Format enthaltenen Kommas obligatorisch.

- Notizen -

ADJUST ($EDIT)

Dieser Befehl verschiebt alle Daten in einem oder mehreren Records nach links oder rechts.

Format:

```
ADJUST  [ / *n      [,m      ] \ ]   / L \   p
          \ ssssss [,tttttt] /     \ R /
```

Varianten und dazugehörige Parameter:

keiner der Parameter n, m, ssssss und tttttt:
Nur der laufende Record wird bearbeitet.

ssssss : Nur der Record mit der S/Z-Nr ssssss wird bearbeitet.

ssssss,tttttt : Die Records von der S/Z-Nr ssssss bis und mit der S/Z-Nr werden bearbeitet.

*n : Nur der Record mit der relativen Position n wird bearbeitet.

*n,m : Nur die Records mit den relativen Positionen n bis m werden bearbeitet.

L : Verschieben nach links.

R : Verschieben nach rechts.

p : Anzahl Stellen, um welche die Daten zu verschieben sind, 1 bis 79.

Bemerkungen:

Bei den Varianten mit Von-bis-Begrenzung (*n,m und ssssss,ttttt) muss nur die erste, nicht aber die zweite Positions- bzw. S/Z-Nummer im EWFvorhanden sein. Die Verarbeitung wird beendet, wenn ein Record mit einer grösseren Nummer als m bzw. ttttt oder das Dateiende erreicht wird.

Der ganze Recordinhalt wird um p Positionen nach links oder rechts verschoben.
Hinter der Verschiebung werden Leerstellen eingefügt.
Daten, die über den Recordanfang oder über die Spalte 80 hinausgeschoben würden, gehen verloren.

Wurde mit dem Befehl ASSIGN CPLMOVE = TRUE festgelegt, wird der erste auf die bearbeiteten Records folgende Record zum laufenden und auf dem Bildschirm ausgegeben (vergleiche Befehl ASSIGN).
Im Fall von CPLMOVE = FALSE bleibt die Position des laufenden Records unverändert.

Beispiele: (Bildschirm)

a) Verschieben der Daten im laufenden Record um 4 Stellen nach rechts:

AD R 4 (NL)

b) Verschieben aller Daten vom 100. bis zum 120. Record um 10 Stellen nach links:

AD *100,120 L 10 (NL)

c) Verschieben aller Daten vom Record mit der S/Z-Nummer 100300 bis zur S/Z-Nummer 101450 um 7 Stellen nach rechts:

AD 100300,101450 R 7 (NL)

AGAIN ($EDIT)

Wiederholung des zuletzt ausgeführten Text-Editor-Befehls.

Format:

AGAIN

Beispiel:

Der zuletzt ausgeführte Befehl war: PO -1

AG -> führt PO -1 aus

- Notizen -

ASSIGN ($EDIT)

Dieser Befehl hat verschiedene Varianten:

- Festlegen von Parametern für die Funktion anderer Texteditor-Befehle.
- Zuordnen von Dateien während des Texteditor-Laufes.

Formate zum Festlegen von Parametern für die Funktion anderer Texteditor-Befehle:

```
ASSIGN

ASSIGN  / CPLMOVE  \  [ = / TRUE \ ]
        | RECORDS  |       \ FALSE/
        \ SUPPRESS /

ASSIGN  COLUMN  [ = z ]

ASSIGN  TABCHAR [ = x ]

ASSIGN  DUPCHAR  [ = x ]

ASSIGN  TERMCHAR [ = x ]
```

Format zum Zuordnen von Dateien während des Texteditor-Laufes:

```
ASSIGN FILE= logname physname (...) parameter
```

---> ab logname genau wie im Systembefehl ASSIGN
(vgl. S. 4/ASS/1 ff)

Parameter:

-ohne Parameter:
Für alle Parameter gilt die Default-Angabe (siehe einzelne Parameter).

-CPLMOVE:
TRUE (Default) = Nach jedem Texteditor-Befehl wird der als letztes bearbeitende Record zum laufenden Record.
FALSE Folgende Befehle lassen den laufenden Record unverändert:
AD, CO, DE, DI, IN, MO, RE, PR, SA und ST.

- **RE**CORD:
FALSE (Default) = Mit DISPLAY, STRING oder SEARCH ausgegebene Records erscheinen ohne Angabe ihrer relativen Position.
TRUE Vor jedem mit DISPLAY, STRING oder SEARCH ausgegebenen Record erscheint seine relative Position.

- **SU**PRESS:
FALSE (Default) = Alle mit STRING verän-derten Records werden auf dem Bildschirm ausgegeben.

TRUE Die mit STRING veränderten Records nicht auf dem Bildschirm ausgeben.

- **CO**LUMN:
z = Relative Zeichen-Position im Record, ab welcher SEARCH- und STRING-Befehle den Record-Inhalt untersuchen.
Default-Wert ist 1.

- **TA**BCHAR:
x = Das Zeichen, welches in INSERT-Funktionen als Tabulations-Zeichen verwendet wird.

Default-Zeichen ist @.
Jedes Zeichen der Bildschirm-Tastatur kann als Tabulationszeichen verwendet werden.
Es empfiehlt sich jedoch, nur Zeichen zu verwenden, die nirgends als Daten-Zeichen vorkommen. Das Zeichen # ist nicht geeignet, da es zum Einfügen von Leerzeilen verwendet wird (siehe unter INSERT).

- **DU**PCHAR:
  x = Das Zeichen, welches in INSERT- und CHANGE-Funktionen als Duplizier-Zeichen verwendet wird.
  Default-Zeichen ist ^.
  Es kann jedes Zeichen ausser Punkt, Leerstelle und das Tabulationszeichen angegeben werden.

- **TE**RMCHAR:
  x = Das Zeichen, welches im INSERT-Befehl die Einfügung von Records beendet.
  Default-Zeichen ist ö oder (NL).
  Es kann jedes Zeichen ausser Punkt, Leerstelle und das Duplizierzeichen angegeben werden.

Bemerkungen:

Ein ASSIGN-**Parameter für die Funktion von andern Befehlen** gilt während eines Texteditor-Laufes so lange, bis derselbe Parameter mit einem neuen ASSIGN-Befehl verändert wird.

Parameter-Angaben ohne die Spezifikationen TRUE, FALSE, z oder x setzen den Default-Wert des betreffenden Parameters ein.

Nach dem Start des Texteditors (EX $EDIT) gelten die Default-Angaben für alle Parameter.

Der ASSIGN-Befehl für die **Datei-Zuordnung** (mit FI=...) gilt bis zum Abschluss des Text-Editor-Laufes.

Beispiele:

a) Für alle Parameter soll die Default-Angabe gelten:

```
AS (NL)
```

b) Nach jedem Text-Editor-Befehl soll der zuletzt bearbeitete Record zum laufenden werden:

```
AS CP (NL)      oder
AS CP = T (NL)
```

c) Die Position des laufenden Records im EWF soll durch die Befehle CO, DE, DI, IN, MO, RE, PR, SA und ST nicht verändert werden:

```
AS CP = F (NL)
```

d) Vor jedem mit DISPLAY ausgegebenen Record soll seine relative Position auf dem Bildschirm erscheinen:

```
AS RE = T (NL)
```

e) Die SEARCH- und STRING-Befehle sollen alle Records erst ab Spalte 20 untersuchen:

```
AS CO = 20 (NL)
```

f) Die SEARCH- und STRING-Befehle sollen alle Records wieder ganz von vorn untersuchen:

```
AS CO (NL)      oder
AS CO = 1 (NL)
```

g) Mit STRING veränderte Records nicht auf dem Bildschirm ausgeben:

```
AS SU = T (NL)
```

h) Als Tabulationszeichen im Befehl INSERT soll das Dollar-Zeichen gelten:

```
AS TA = $
```

Weitere Beispiele finden Sie unter den Text-Editor-Befehlen, die durch ASSIGN variiert werden können.

i) Zuordnung der bestehenden Disc-Datei S-BEISP (2) mit dem lognamen B:

```
AS FI= B S-BEISP (2)
```

k) Zuordnung einer neuen Disc-Datei S-NEU (2), 200 Sektoren, mit dem lognamen B:

```
AS FI= B S-NEU (2) NE 200
```

l) Zuordnung des Druckers 1 mit dem lognamen LO:

```
AS FI= LO (1 LP)
```

- Notizen -

CHANGE ($EDIT)

Dieser Befehl sucht einen Record, gibt ihn auf den Bildschirm aus und ermöglicht danach Veränderungen des Records.

Format:

CHANGE [/ssssss\ ] \ *n /	Befehls-Eingabe
X----------X	Ausgabe des Records
Neuer Inhalt (NL)	**Antwort:** Record-Aenderung. Mit dem Zeichen ^ lassen sich Zeichen von der vorangehenden Zeile duplizieren.
oder:	
nur (NL) oder ö(NL) oder \(NL)	Record bleibt **unverändert**

Parameter:

keine Parameter = Bearbeitung des laufenden Records

ssssss = Bearbeitung des Records mit der S/Z-Nummer ssssss.

*n = Bearbeitung des Records mit der relativen Position n.

Beispiel: (Bildschirm)

Verändern eines Teils des 200. Records:

CH *200 (NL)	Eingabe
100300 MOVE F22 TO X22.	Ausgabe
^^^^^^^^^^^^^^^^^^^^X-22.	Eingabe, ^ = duplizieren
100300 MOVE F22 TO X-22.	Ausgabe neuer Zustand

- Notizen -

CONCAT ($EDIT)

Dieser Befehl liest eine Text-Editor-kompatible Datei oder einen Teil derselben ins EWF. Die gelesenen Records werden nach dem laufenden Record im EWF eingefügt.

Format:

a) Für Dateien mit variabler Recordlänge:

```
CONCAT  a     [, / ssssss  [,tttttt] \ ]
                 \ *[n]    [,m]      /
```

b) Für Dateien mit fixer Recordlänge:

```
CONCAT  a     (r,b)  [, /ssssss  [,tttttt] \ ]
                        \  *[n]   [,m]     /
```

a = Logischer Name (logname) aus dem ASSIGN-Befehl für die Eingabe-Datei. Meistens wird dafür A oder ein anderer Buchstabe verwendet.

Nur für Format b) gelten:

r : Recordlänge der Eingabe-Datei
b : Blocklänge der Eingabe-Datei

Varianten und dazugehörige Parameter

Ausser a keine Parameter :	Die **ganze** Eingabedatei wird eingelesen.
ssssss	Aus der Eingabedatei werden nur die Records ab dem Record mit der S/Z-Nummer ssssss übernommen.
ssssss,tttttt	Aus der Eingabedatei werden nur die Records von der S/Z-Nummer ssssss bis und mit der S/Z-Nummer tttttt übernommen.

*n	:	Aus der Eingabedatei werden nur die Records ab der relativen Position n übernommen.
*n,m	:	Aus der Eingabedatei werden nur die Records von der relativen Position n bis und mit Position m übernommen.
*,m	:	Aus der Eingabedatei werden nur die Records bis zur Position m übernommen, d.h. die ersten m Records.

Bemerkungen:

Für jede mit CONCAT zu lesende Datei muss vor dem Start des Text-Editors mit EX $EDIT ein entsprechender ASSIGN-Befehl ablaufen.

Im Falle von CPLMOVE = TRUE wird der letzte eingefügte Record zum laufenden und am Bildschirm ausgegeben (Normalfall).
Gilt CPLMOVE = FALSE, verändert der CONCAT-Befehl die laufende Record-Position nicht (vgl. Texteditor-Befehl ASSIGN).

Die S/Z-Nummer der eingelesenen Records bleiben unverändert. Dadurch können im EWF Seiten/Zeilennummer-Ueberschneidungen, Verdoppelungen oder unerwünschte Sequenzen entstehen. Mit dem Befehl RESEQUENCE lassen sich in solchen Fällen neue S/Z-Nummern generieren.

Beispiele: (Bildschirm)

Für alle Beispiele wird angenommen: CPLMOVE = TRUE

a) Eine ganze Datei mit variabler Recordlänge einlesen:

CO A (NL)	Eingabe
401500 STOP RUN.	Ausgabe des letzten eingelesenen Records.

b) Nur den 10. bis 50. Record einlesen:

CO A, *10,50 (NL)	Eingabe
200760 EXIT.	Ausgabe des letzten übernommenen Records.

c) Eine ganze Datei mit fixer Recordlänge 80 einlesen:

CO A (80,480)	Eingabe
ENDE xxxxxxxx	Ausgabe des letzten Records.

- Notizen -

DCL ($EDIT)

Dieser Befehl legt die Länge der S/Z-Nummer fest.
Ist diese 6-stellig, wird der DCL-Befehl nicht benötigt.

Format:

DCL s

Varianten:

ohne Parameter : Die S/Z-Nummer ist 6-stellig

s : Länge der S/Z-Nummer, 1 bis 8 Stellen

Bemerkung:

Enthält eine Datei Records mit unterschiedlich langen S/Z-Nummern, finden Texteditor-Befehle mit S/Z-Nummer als Suchbegriff nicht in allen Fällen die gesuchten Records.

Beispiele:

a) Die ersten 6 Stellen jedes Records sind die S/Z-Nummer, entsprechend dem COBOL-Format:

DC 6 (NL) oder: **DC (NL)**

b) Die ersten 4 Stellen jedes Records sollen als S/Z-Nummer gelten:

DC 4 (NL)

- Notizen -

DELETE ($EDIT)

Dieser Befehl löscht einen oder mehrere Records aus dem EWF.

Format:

```
DELETE  [ / ALL                \ ]
          | ssssss [,tttttt] |
          \ *n     [,m     ] /
```

Varianten und dazugehörige Parameter:

keine Parameter	:	Nur der laufende Record wird gelöscht.
ssssss	:	Nur der Record mit der S/Z-Nr ssssss wird gelöscht.
ssssss,tttttt	:	Die Records von der S/Z-Nr ssssss bis und mit der S/Z-Nr werden gelöscht.
*n	:	Nur der Record mit der relativen Position n wird gelöscht.
*n,m	:	Nur die Records mit den relativen Positionen n bis m werden gelöscht.
ALL	:	Löschen aller Records im EWF.

Bemerkungen:

Bei den Varianten mit Von-bis-Begrenzung (*n,m und ssssss,tttttt) muss nur die erste, nicht aber die zweite Positions- bzw. S/Z-Nummer im EWF vorhanden sein. Die Verarbeitung wird beendet, wenn ein Record mit einer grösseren Nummer als m bzw. tttttt oder das Dateiende erreicht wird.

Wurde mit dem Befehl ASSIGN CPLMOVE = TRUE festgelegt, wird der erste auf die gelöschten Records folgende Record zum laufenden und auf dem Bildschirm ausgegeben (vergleiche Befehl ASSIGN).
Im Fall von CPLMOVE = FALSE bleibt die Position des laufenden Records unverändert.

Beispiele: (Bildschirm)

Für alle Beispiele wird angenommen: CPLMOVE = TRUE (Normalfall).

a) Löschen des 20. Records:

DE *20 (NL)	Eingabe
720600 ADD 2 TO X1.	Ausgabe des vorher 21. Records, der nun der 20. geworden ist.

b) Löschen der Records von S/Z-Nummer 200000 bis 203630:

DE 200000,203630 (NL)	Eingabe
204000 UP16.	Ausgabe des ersten Records nach dem gelöschten Bereich.

DISPLAY ($EDIT)

Mit diesem Befehl können einer oder mehrere EWF-Records auf den Bildschirm ausgegeben werden.

Format:

```
DISPLAY    [ / ALL                  \ ]
             | ssssss [,tttttt ]    |
             \ *n     [,m      ]   /
```

Varianten:

-	Keine Parameter:	Der laufende Record wird ausgegeben.
-	ssssss ssssss,tttttt *n *n,m	Diese Parameter sind wie die entsprechenden Angaben im Befehl DELETE zu verstehen. Statt "gelöscht" ist jedoch "ausgegeben" einzusetzen.
-	**ALL**	Ausgabe aller Records im EWF. Nach jedem vollen Schirm stoppt die Ausgabe und erlaubt den Abbruch der Ausgabe.

Bemerkungen:

Je nach den Parametern, die mit dem Texteditor-Befehl ASSIGN festgelegt wurden, gilt:

Bei **CP**LMOVE = **TRUE**	:	Der letzte ausgegebene Record wird zum laufenden.
CPLMOVE = **FALSE**	:	Die laufende Record-Position wird durch die Ausgabe nicht verändert.
Bei **RE**CORD = **TRUE**	:	Vor jedem ausgegebenen Record erscheint seine relative Positionsnummer.
RECORD = **FALSE**	:	Ausgabe der Records ohne ihre relativen Positionen.

Um alle EWF-Records auf den Bildschirm zu bringen, sind folgende zwei Varianten anwendbar:

- DI AL

- DI *1,9999 — statt 9999 kann irgendeine Zahl eingesetzt werden, die mindestens so gross ist, wie die Record-Anzahl im EWF, höchstens jedoch 65535.

Records die 80 Zeichen lang sind, ergeben auf dem Bildschirm eine Leerzeile unter jedem Record.

Beispiele: (Bildschirm)

a) Ausgabe des 3. bis 6. Records aus dem EWF:

Annahmen: CPLMOVE=TRUE, RECORD=FALSE(Normalfall).
Die laufende Record-Position sei 22.

```
22C?DI 3,6 (NL)        Eingabe: DI *3,6

xxxxxxxxxxxxxxxx
xxxxxxxxxxxxx          Ausgabe des 3. bis 6.
xxxxxxxxxxxxxxxxx      Records
xxxxxxxxxx
6C?                    Ausgabe: Der 6. Record ist
                       der laufende geworden.
```

b) Ausgabe des Records mit der Seiten/Zeilen-Nummer 220730:

Annahmen: CPLMOVE=FALSE, RECORD=TRUE,
Die laufende Record-Position sei 10

```
10C?DI 220730 (NL)     Eingabe: DI 220730

   127                 Ausgabe: rel. Position ist
220730 xxxxxxxxxx                127 und
                                 Record-Inhalt

10C?                   Ausgabe: Der 10.
                       Record ist immer noch der
                       laufende.
```

c) Ausgabe aller Records im EWF:

Annahmen: CPLMOVE=TRUE, RECORD=TRUE,
Die laufende Record-Position sei 10.

```
10C?DI AL (NL)          Eingabe: DI AL
    1
xxxxxxxxxxxxxx          Ausgabe:
    2                   alle Records.
xxxxxxxxxxxxxxx         Ueber jedem Record steht
    3                   seine relative Position.
xxxxxxxxxx
    4
xxxxxxxxxxxxx
   usw
```

Nach jedem vollen Schirm erscheint die Meldung:
** N/L TO CONTINUE, X TO CANCEL
** ENTER RESPONSE
Nach NEWLINE folgt die nächste Seite, mit X wird die DISPLAY- Funktion abgebrochen.

```
nnnC?                   Ausgabe: Position des
                        letzten ausgegebenen
                        Records.
```

- Notizen -

ERASE ($EDIT)

Dieser Befehl löscht den Bildschirm.

Format:

ERASE

Bemerkung:

Nach der Ausführung steht auf dem Schirm nur noch der laufende Record und die Meldung: n C?

Beispiel:

Den Schirm löschen:

ER (NL)	Eingabe
xxxxxxxxxxxxxxx n C?	Ausgabe: laufender Record, Nummer des laufenden Records.

- Notizen -

FIND ($EDIT)

Dieser Befehl sucht einen Record im EWF, macht ihn zum laufenden und gibt ihn auf dem Bildschirm aus.

Format:

```
      FIND      /ssssss\
                \ *n   /
```

Parameter

ssssss = Ausgabe des Records mit dieser S/Z-Nr.

*n = Ausgabe des Records mit der relativen Position n.

Beispiel: (Bildschirm)

FI 111000 (NL) Eingabe

111000 WRITE ZEILE1.Ausgabe

Bemerkung

Wenn AS CP=T gilt, ist der Befehl FIND identisch mit dem entsprechenden DISPLAY-Befehl.

Wenn AS CP=F gilt, wird der FIND-Befehl benötigt,um die laufende Position im EWF zu verändern.

GRID ($EDIT)

Gibt einen Raster aus.

Format:

GRID

Bemerkung:

Der Raster wird oberhalb des laufenden Records angezeigt.

Beispiel:

```
GR (NL)        Eingabe
               Ausgabe:

....+....1....+....2....+..usw. bis ..+....8
```

INSERT ($EDIT)

Dieser Befehl ermöglicht das Einfügen eines Records oder einer zusammenhängenden Recordfolge ins EWF.
Die neuen Records werden nach dem laufenden Record eingefügt.

Nach Eingabe des INSERT-Befehls wird am Schirm ein Raster mit den Positionen 1 bis 80 ausgegeben. Jede darauf eingegebene Zeile ergibt einen Record. Records, die nicht die ganze Zeile füllen, werden mit NEWLINE abgeschlossen.

Zum Abschluss der einzufügenden Record-Gruppe gibt man anstelle eines Recordinhaltes am Zeilenanfang nur (NL) oder ö (NL) oder \ (NL) ein.

Formate:

a) **INSERT * [D]**

b) **INSERT [D]**

Parameter

mit * (Format a) : Das erste eingebene Zeichen einer Zeile ergibt die Spalte 1 im Record. Somit können ganze Records eingegeben werden.

ohne * (Format b) : Das erste eingegebene Zeichen einer Zeile ergibt die erste Spalte nach der S/Z-Nummer gemäss DCL-Befehl. Die Spalten davor bleiben in allen eingefügten Records leer.
Diese Form wird besonders für die Eingabe von COBOL-Befehlen ohne S/Z-Nummer verwendet.
Der Cursor darf nicht in die S/Z-Nr. zurückgesetzt werden (ausser mit Tabulatoren).

mit D : Bei der Eingabe lassen sich mit dem Zeichen ^ Daten vom vorangehenden Record duplizieren.

Wichtig:

Falls am Zeilenanfang Leerstellen einzusetzen sind, müssen diese mit der Leer-Taste eingegeben werden. Die Taste -> ergibt Zufalls-Inhalte in den übersprungenen Stellen!

Im Fall von CPLMOVE = TRUE wird der letzte eingefügte Record zum laufenden. Seine Position ist aus der Aufforderung zum nächsten Befehl ersichtlich.
Falls mit ASSING CPLMOVE = FALSE festgelegt wurde, bleibt die laufende Record-Position unverändert (siehe Texteditor-Befehl ASSIGN).

Beispiele:

a) Eingabe dreier Records ab Spalte 1, mit Dupliziermöglichkeit:

Annahme: CPLMOVE=TRUE, Laufender Record : 100

xxxxxxxxxx	Ausgabe des laufenden Records
100C?**IN*D (NL)**	Eingabe des Befehls
....+....1....+....2...	Ausgabe des Rasters
AS A D234 (2) (NL)	Eingabe des 1. Records
^^^B M234 (MT)NE(NL)	Eingabe des 2. und 3.
^^^D (LP) (NL)	Records, deren erste Zeichen dupliziert werden.
(NL)	Abschluss der Eingabe.
AS D (LP)	Ausgabe des letzten ein-
103C?	gegebenen Records, der zugleich der laufende geworden ist.

b) Eingabe eines Records mit leerer S/Z-Nummer in Spalte 1-6.

Annahme: CPLMOVE = FALSE, laufender Record: 150
DCL = 6.

150C?**IN (NL)**	Eingabe des Befehls-**ohne ***
....+....1....+....2....+..	Ausgabe des Rasters
A5. MOVE "X" TO W22.(NL)	Eingabe des Records (1.Zeichen=**Spalte 7)**
(NL)	Abschluss der Eingabe
xxxxxxxxxxxxxxxx	Ausgabe des laufenden Records (unverändert,
150C?	da CPLMOVE=FALSE)

---

oder:

150C?**IN* (NL)**	Eingabe des Befehls **mit ***
....+....1....+....2....+..	Ausgabe des Rasters
A5. MOVE "X" TO W22.(NL)	
|	Eingabe des Records
6 Leerstellen eingeben	(1.Zeichen= **Spalte 1)**
(NL)	Abschluss der Eingabe
xxxxxxxxxxxxxxxx	Ausgabe des laufenden Records
150C?	(unverändert,da CPLMOVE=FALSE)

TABULATION

In den Daten-Zeilen, die nach dem INSERT-Befehl eingegeben werden, kann man wie folgt tabulieren:

- Die Tabulatoren müssen mit dem Texteditor-Befehl TAB festgelegt werden. Siehe unter Befehl TAB.

- Das Zeichen, mit dem auf einen Tabulator gesprungen werden kann, lässt sich mit dem Texteditor-Befehl ASSIGN wählen. Siehe unter Befehl ASSIGN.

  Die gesetzten Tabulatoren-Positionen und das Tabulations-Zeichen sind im Raster über der Dateneingabe ersichtlich, z.B:

```
....+....1.%..+..%.2...%+....3.%..+....4....+..
```

  Die Tabulatoren sind auf die Positionen 12, 18, 24, 32 gesetzt, das Tabulations-Zeichen ist in diesem Beispiel das %-Zeichen.

Jedesmal, wenn ein Tabulations-Zeichen in der Daten-Eingabe steht, werden die nachfolgenden Daten auf die nächste Tabulations-Position gesetzt.

Zwei aufeinanderfolgende Tabulations-Zeichen bedeuten, dass die übernächste Tabulations-Position angesprungen werden soll, usw.

Ist die gesamte Anzahl Tabulations-Zeichen in einer Eingabezeile grösser als die Anzahl der gesetzten Tabulatoren, werden die überzähligen Tabulations-Zeichen ignoriert.

Beispiele:

a) Annahmen: Gesetzte Tabulatoren: 8, 16, 32, 48.
Tabulationszeichen: @

Bildschirm-Dialog:

IN* (NL)	Eingabe
....+..@.1....+@...2....	Ausgabe des Rasters
@1@W4@PIC X. (NL)	Eingabe der Daten-zeile mit dem Tabulations-Zeichen@
(NL)	Abschluss

ergibt als eingefügten Record:

```
Spalte 8      16       32
-------------------------------
|       1     W4       PIC X.  |
-------------------------------
```

b) Annahmen: Gesetzte Tabulatoren: 8, 16,28
Tabulations-Zeichen: %
S/Z-Nummer: 6 Stellen

Bildschirm-Dialog:

IN (NL)	Eingabe
....+..%.1....+%...2....+..%	Ausgabe des Rasters
***%%DATUM-PRUEFUNG (NL)**	Daten-Eingabe

ergibt als eingefügten Record:

```
Spalte 7       28
---------------------------------
|      *       DATUM-PRUEFUNG   |
---------------------------------
```

c) Annahmen: Gesetzte Tabulatoren: 8, 12
Tabulations-Zeichen: %

Bildschirm-Dialog:

IN*D (NL)	Eingabe
....+..%.1.%..+....2....+	Ausgabe des Rasters
%H2.%GO TO%H30. (NL)	Daten-Eingabe
%^3^%MOVE W1 TO W2. (NL)	
(NL)	Abschluss

ergibt als eingefügte Records:

```
Spalte 8   12
-------------------------- Das 3. %-Zeichen wurde
|      H2. GO TOH30.     | ignoriert!
--------------------------
--------------------------
|      H3. MOVE W1 TO W2.| Das "H" und der
-------------------------- Punkt werden
                           dupliziert.
```

Bemerkungen zum Duplizieren:

Bei Verwendung von Tabulations-Zeichen kann nur unter den folgenden Bedingungen vom vorangehenden Record dupliziert werden:

- Tabulationszeichen lassen sich nicht duplizieren
- Nach der Eingbe des ersten Tabulations-Zeichens in einem Record können nur Daten dupliziert werden, die auf dem Schirm genau über dem ^-Zeichen stehen.
- Anzahl und Position der Tabulations-Zeichen muss in beiden Records gleich sein.

EINFUEGEN VON LEEREN RECORDS

Wird innerhalb einer INSERT-Recordgruppe statt eines neuen Records als einzige Eingabe pro Zeile

```
#Sz
```

eingegeben, werden z leere Records ins EWF eingefügt, maximal 20.

Beispiel

EWF-Inhalt vorher:

```
-------------------------
|  P2.                  |
|     MOVE W1 TO W2.    |<--- laufender Record
|                       |
|  P3.                  |
|     IF W-4 = 6 GO P5. |
-------------------------
```

Eingabe:

```
IN* (NL)
#S2 (NL)
******* TEST 4 (NL)
(NL)
```

EWF-Inhalt nachher:

```
-------------------------
|  P2.                  |
|     MOVE W1 TO W2.    |
|   (leer)              |<--- 2 leere Records
|   (leer)              |
|***TEST 4              |<--- eingefügter Record
|  P3.                  |
|     IF W-4 = 6 GO P5. |
|                       |
-------------------------
```

- Notizen -

MOVE ($EDIT)

Dieser Befehl überträgt EWF-Records an eine neue Position im EWF.
Die übertragenen Records werden an der bezeichneten Stelle zwischen die bereits bestehenden eingefügt.
Die ursprünglichen Records lassen sich bei Bedarf löschen.

Formate:

a) Uebertragen von Records mit bestimmten **relativen Positionen** an eine gewünschte relative Position.

   MOVE [DELETE,] ***n** [,m] **-p**

b) Uebertragen von Records mit bestimmten **Seiten/Zeilen-Nummern:**

   MOVE [DELETE,] **ssssss** [,tttttt] **-uuuuuu** [,i]

Parameter

DELETE = Die ursprünglichen Records werden gelöscht.

n = Relative Position des ersten zu übertragenden Records, falls mehrere zu übertragen sind.

m = Relative Position des letzten zu übertragenden Records, falls mehrere zu übertragen sind.

p = Relative Position, nach welcher die übertragenen Records einzufügen sind.

ssssss = S/Z-Nummer des ersten zu übertragenden Records.

tttttt = S/Z-Nummer des letzten zu übertragenden Records, falls mehrere Records zu übertragen sind.

uuuuuu = S/Z-Nummer des Records, nach welchem die übertragenen Records einzufügen sind.

i = Erhöhungswert der S/Z-Nummer von Record zu Record. Fehlt dieser Parameter, wird 10 angenommen. Entstehen Ueberlappungen der S/Z-Numerierung, erfolgt eine Warnung auf dem Bildschirm.

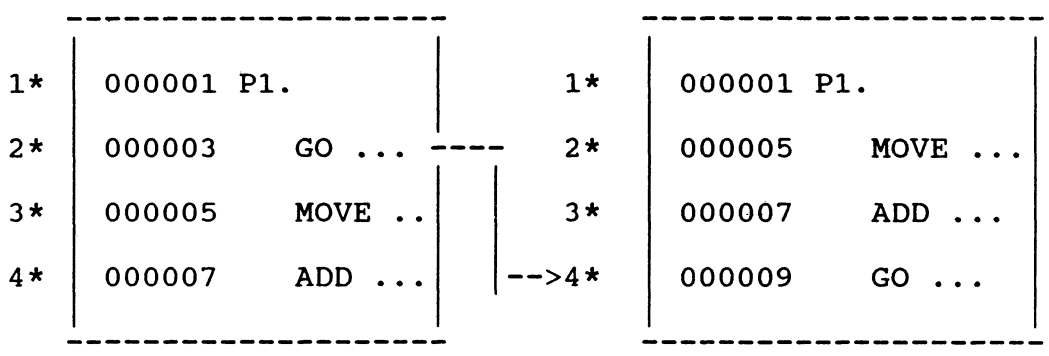
Bemerkungen

Falls CPLMOVE = TRUE gilt, wird der letzte übertragene Record an seiner neuen Position zum laufenden Record. Wurde mit dem Befehl ASSIGN CPLMOVE = FALSE festgelegt, bleibt die Position des laufenden Records durch den MOVE-Befehl unverändert (Vergleiche Befehl ASSIGN).

Beispiele (Bildschirm)

Annahme: DCL = 6 (Normalfall)

Bestehender EWF-Inhalt: Gewünschter EWF-Inhalt:



Lösungen:

unter der Annahme von CPLMOVE = TRUE:

a) Den 2. Record an die Stelle nach dem 4. schieben
Nachher die S/Z-Nummer ändern.

MO DE, *2-4 (NL)	Eingabe
000003 GO P4.	Ausgabe des laufenden Records.
CH (NL)	Eingabe
000003 GO P4.	Ausgabe
^^^^^9^^^^^^^^(NL)	Eingabe neue S/Z-Nr.

b) Denselben Record mit Angabe der Seiten/-Zeilen-Nummer verschieben und zugleich die S/Z-Nummer neu einsetzen:

MO DE, 000003-000007,2	Eingabe
000009 GO P4.	Ausgabe des laufenden Records

c) Der 3. und 4. Record soll unverändert auch nach dem 1. eingefügt werden:

Annahme: CPLMOVE = FALSE.
Der laufende Record sei *1.

Bestehender EWF-Inhalt:

```
    -------------------------
    |                       |
1*  | 000001 P1.            |
    |                       |
2*  | 000003    GO ...      |
    |                       |
3*  | 000005    MOVE ...    |
    |                       |
4*  | 000007    ADD ...     |
    |                       |
    -------------------------
```

MO *3,4-1 (NL) Eingabe

000001 P1. Ausgabe des laufenden Records, unverändert.

EWF-Inhalt nach dem Befehl:

```
    -------------------------
    |                       |
1*  | 000001 P1.            |
    |                       |
2*  | 000005    MOVE ...    |  }
    |                       |  }<-----
3*  | 000007    ADD ...     |  }      |
    |                       |         | eingeschoben
4*  | 000003    GO  ...     |         |
    |                       |         | (verdoppelt)
5*  | 000005    MOVE ...    |  }      |
    |                       |  }------
6*  | 000007    ADD ...     |  }
    |                       |
    -------------------------
```

POSITION ($EDIT)

Mit diesem Befehl wird der laufende Record im EWF um eine bestimmte Anzahl Records vor- oder rückwärts positioniert.
Der neue laufende Record wird auf dem Bildschirm ausgegeben.

Format:

```
POSITION  / n  \
          \ -n /
```

Parameter:

n = vorwärts positionieren um n Records.

-n = rückwärts positionieren um n Records.

Bemerkung:

Statt PO 1 kann auch nur die NEWLINE-Taste gedrückt werden, um den nächstfolgenden Record auszugeben.

Beispiel: (Bildschirm)

PO -1 (NL)	Eingabe
220710 MOVE 0 TO S1.	Ausgabe des vorangehenden Records.

- Notizen -

PRINT ($EDIT)

Dieser Befehl druckt einen oder mehrere EWF-Records auf einem Drucker aus.

Jeder Record belegt eine Zeile.
Eine Seite umfasst soviele Zeilen, wie als Seitenhöhe bei der System-Generierung festgelegt wurden.
Am linken Blattrand werden die relativen Positionen der Records ausgedruckt.

Format:

```
PRINT    [ / ALL                 \ ]
           | ssssss  [,tttttt]   |
           \ *n      [,m ]       /
```

Parameter:

Die Parameter haben dieselben Funktionen wie im DISPLAY- und im DELETE-Befehl (siehe dort).

Bemerkungen:

Dem Text-Editor muss ein Drucker oder ein Spoolfile mit dem lognamen **LO** zugeordnet sein, z.B.

AS **LO** (LP)	direkter Druck oder Auto-Spooling
oder	
AS **LO** physname(n) NE s SP	manuelles Spoolfile

Falls CPLMOVE = TRUE gilt, wird der letzte gedruckte Record zum laufenden und auf dem Bildschirm ausgegeben. Wurde mit dem Texteditor-Befehl ASSIGN CPLMOVE = FALSE festgelegt, bleibt die laufende Record-Position unverändert (siehe Texteditor-Befehl ASSIGN).

Beispiele (Bildschirm):

Annahme: CPLMOVE = TRUE

a)Ausdrucken der ersten 50 Records

PR *1,50 (NL)	Eingabe
090600 OPEN INPUT F1.	Ausgabe des 50. Records.

b)Ausdrucken aller Records ab (inklusive) S/Z-Nummer 600000):

PR 600000,999999 (NL)	Eingabe
920600 EXIT.	Ausgabe des letzen Records.

QUIT ($EDIT)

Mit diesem Befehl wird der Text-Editor-Lauf abgeschlossen.
Das EWF bleibt unverändert erhalten. Es kann nach einer neuen Zuteilung (AS EWF-Befehl) und dem Start mit EX $EDIT,RE weiterverarbeitet werden.

Format:

QUIT

Bemerkungen:

Wurden seit dem Start des Editors oder seit dem letzten SAVE-Befehl Aenderungen im EWF gemacht erfolgt ab ITX Rel.2.2 folgende Warnung:

```
**TEXT MAY HAVE BEEN MODIFIED**
**DO YOU WANT TO ENTER THE SAVE COMMAND? (YES ODR NO)
**ENTER RESPONSE:
```

Um den SAVE nachzuholen: Y oder YES eingeben,
um ohne SAVE abzuschliessen: N oder NO eingeben.

Beispiel:

Mit dem EWF-Inhalt des letzten Laufes weiterarbeiten:

```
-------------------------------------------------
| Hier wird im EWF gearbeitet.                  |
| Z.B. wegen eines Fehlers im AS des Files lässt|
| sich der Befehl SA A nicht ausführen.         |
|                                               |
| QU                 Text-Editor abschliessen   |
|                                               |
| AS EWF EWFxx (n)   Dasselbe EWF neu zuteilen  |
|                                               |
| AS A .......,OW    Das mit SA A zu beschrei-  |
|                    bende File zuteilen, own ! |
|                                               |
| EX $EDIT,RE        Text-Editor mit dem alten  |
|                    EWF-Inhalt starten, mit RE!|
|                                               |
| DI *n,m            Gegebenenfalls Kontrolle   |
|                                               |
| SA A               Den vorher nicht möglichen |
|                    SA auszuführen             |
|                                               |
| QU                 Text-Editor abschliessen   |
-------------------------------------------------
```

Die letzten 2 Befehle könnten auch auf einer Zeile eingegeben werden, mit einem Punkt getrennt:

```
SA A.QU
```

RESEQUENCE ($EDIT)

Dieser Befehl versieht eine geschlossene Recordgruppe oder alle Records im EWF mit neuen, aufsteigenden S/Z-Nummern.

Er ist nur für Dateien sinnvoll, die in den ersten Stellen jedes Records keine andern Daten als eine S/Z-Nummer aufweisen können, insbesondere für COBOL-Source-Programme.

Formate:

a) Das ganze EWF mit neuen S/Z-Nummern versehen:

RESEQUENCE **ssssss** [,i]

b) Nur einen Teil des EWF:

RESEQUENCE **tttttt,uuuuuu-ssssss** [,i]

Parameter:

ssssss = erste neu einzusetzende S/Z-Nummer

i = Erhöhungswert der S/Z-Nummer von Record zu Record. Fehlt dieser Parameter, wird um 10 erhöht.

tttttt = S/Z-Nummer des ersten zu verarbeitenden Records.

uuuuuu= S/Z-Nummer des letzten zu verarbeitenden Records.

Bemerkungen:

Die Länge der S/Z-Nummer ist vom gültigen Texteditor-Befehl DCL abhängig. Sie kann dort 1 bis 8-stellig gewählt werden. Ohne Verwendung des DCL-Befehls ist die S/Z-Nummer 6-stellig.

Die ersten 1 bis 8 Stellen (je nach DCL-Parameter) aller betroffenen Records werden durch die Neunumerierung überdeckt, unabhängig vom alten Inhalt. Daten dahinter bleiben unberührt.
Entstehen durch eine Neunumerierung S/Z-Nummer-Ueberlappungen, erscheint eine entsprechende Warnung auf dem Bildschirm.

Der letzte verarbeitete Record wird als neuer laufender Record auf dem Bildschirm ausgegeben.

Beispiele (Bildschirm)

Annahme: DCL = 6 (Normalfall).

a) Neu-Numerierung des ganzen EWF, erste S/Z-Nr.=100000, Erhöhungswert = 30.

Ausgangslage des EWF:

```
200010 P1.
200050     MOVE ...
           ADD  ...
100900 P9.
100910     EXIT.
```

```
RE 100000,30 (NL)        Eingabe

100120 EXIT              Ausgabe des letzten
                         Records.
```

ergibt als Resultat:

```
100000 P1.
100030    MOVE ...
100060    ADD ...
100090 P9.
100120    EXIT.
```

b) Die Records von S/Z-Nr. 230055 bis 230099 neu numerieren:
Erste einzusetzende S/Z-Nr. = 330000,
Erhöhungswert = 50:

```
RE 230055,230099-330000,50
```

Falls das EWF 5 Records entthält, erhalten die Zeilen die S/Z-Nummern
330000, 330050, 330100, 330150 und 330200.

- Notizen -

SAVE ($EDIT)

Dieser Befehl schreibt den gesamten EWF-Inhalt sequentiell auf eine Datei.
Die Datei wird sequentiell organisiert und erhält den Typ TXT. Das EWF bleibt unverändert erhalten.

Formate:

a) Schreiben variabler langer Records:

SAVE **a** [(,b)]	Zeilen, die kürzer als 80 Stellen eingegeben wurden, werden als Records mit der eingegebenen Länge geschrieben.

b) Schreiben fixer Records:

SAVE **a (r,b)**	Von jeder Zeile werden die ersten r Stellen als je 1 Record auf die Datei geschrieben. Enthält das EWF Zeilen, die länger als r Stellen sind, werden die überzähligen Stellen abgeschnitten.

Parameter:

a = Logischer Dateiname (logname) der Ausgabe-Datei, gemäss ASSIGN-Befehl.
Meistens verwendet man dafür A oder einen anderen Buchstaben.

r = Gewünschte fixe Recordlänge, höchstens 80.

b = Blocklänge. Bei fixer Recordlänge muss sie ein ganzes Vielfaches der Recordlänge r darstellen. Sie darf höchstens 512 Bytes betragen.

Bemerkungen:

Wird eine neue Ausgabe-Datei geschrieben, muss diese mit einem ASSIGN-Befehl mit NE-Parameter zugeteilt sein.

Disc- und Kassetten-Dateien können mit dem SAVE-Befehl auch überschrieben werden.
Zu überschreibende Disc-Dateien müssen als "own" zugeteilt sein (PR-Parameter beim Neuaufbau oder OW-Parameter im ASSIGN-Befehl).
Eine bestehende Datei wird durch den SAVE-Befehl vollständig überschrieben.
Sie wird ungeachtet ihrer früheren Organisation sequentiell organisiert.

Reicht die auf Disc reservierte Dateigrösse für die Aufnahme des EWF-Inhaltes nicht aus, wird die SAVE-Funktion mit der folgenden Meldung abgebrochen, sobald die Datei voll ist:

** INSUFFICIENT SPACE IN FILE **

Das EWF bleibt dabei unverändert erhalten.

Falls CPLMOVE = TRUE gilt, wird nach dem SAVE-Befehle der letzte Record im EWF zum laufenden.
Wurde mit dem Texteditor-Befehl ASSIGN CPLMOVE = FALSE festgelegt, bleibt die Position des laufenden Records unverändert (vergleiche Texteditor-Befehl ASSIGN).

Beispiele: (Bildschirm)

a) variable Recordlänge:

SA B (NL)

b) fixe Recordlänge 50, Blocklänge 500:

SA B (50,500) (NL)

SEARCH ($EDIT).

Dieser Befehl sucht im EWF einen Record, der eine bestimmte Zeichenfolge aufweist.

Der Suchvorgang beginnt mit dem laufenden Record und wird sequentiell fortschreitend ausgeführt.

Alle diese Records werden ab der Spalte untersucht, die mit ASSIGN CO= festgelegt wurden. Ab jener Spalte wird die gesuchte Zeichenfolge irgendwo im Record erkannt (Vergleiche Texteditor-Befehl ASSIGN).

Format:

```
SEARCH      'zeichenfolge'  [/ ,n \]
                             | *d |
                             \,ALL/
```

Parameter:

zeichenfolge = Eine Folge von mindestens einem ASCII-Zeichen. Die Zeichenfolge muss durch Hochkommas oder durch zwei gleiche andere Zeichen begrenzt werden.
SE 'F1' lässt sich auch schreiben als: SE AF1A, SE XF1X oder SE &F1&.

ohne die Parameter ,n und *d:
Das EWF wird maximal bis zum Ende abgesucht.
Beim ersten Record mit der Zeichenfolge wird das Absuchen beendet.

,n = Maximale Anzahl Records, die nach der "Zeichenfolge" abgesucht werden.
Beim ersten Record mit der Zeichenfolge wird das Absuchen beendet.
Enthält keiner der untersuchten Records eine gesuchte Zeichenfolge, wird die Suchfunktion am EWF-Ende beendet.

*d = Das wievielte Auftreten der "Zeichenfolge", bei dem der Text-Editor anhalten soll.
*3 bedeutet somit: Erst beim 3. Auftreten der Zeichenfolge ab der laufenden Zeile anhalten.
Beim ersten Record mit der Zeichenfolge wird das Absuchen beendet.

,ALL = Ab dem laufenden Record wird das EWF bis zum Ende nach der "Zeichenfolge" abgesucht.
Alle Records ab dem laufenden mit der Zeichenfolge werden ausgegeben.
Empfehlung:
Vorher AS RE=T festlegen, damit die Zeilennumern der Records ausgegeben werden.

Bemerkungen:

Der beim Ende der SEARCH-Funktion als letztes bearbeitete Record wird zum laufenden und am Bildschirm ausgegeben.

Beispiele (Bildschirm)

Annahme: COLUMN = 1

Zustand des EWF für alle Beispiele:

```
    ----------------------
1*  |  AS A DF1 (1)       |
2*  |  AS D (LP)          |  <-- laufender Record, 2*
3*  |  AS C AF1(2)        |
4*  |  EX P-F1  (0)       |
5*  |  DI 'ENDE F1 ' (0)  |
    ----------------------
```

a) SE 'F1' (NL) — Eingabe: unbegrenzt suchen

AS C **AF1**(2) — Ausgabe (Record 3)

b) SE 'P' — Eingabe: unbegrenzt suchen

AS D (L**P**) — Ausgabe: derselbe Record enthält ein 'P'.

c) SE 'F1 ' — Eingabe unbegrenzt suchen

EX P-**F1** (0) — Ausgabe (Record 4)

d) SE 'DI' ,2 — Eingabe: Max. 2 Records absuchen

ITEM NOT FOUND
AS C AF1(2) — Ausgabe: der letzte abgesuchte Record

e) SE 'F' *3 — Eingabe: 3. Auftreten suchen

DI 'ENDE **F1**' (0) — Ausgabe: Der Record, wo zum 3. Mal ab dem laufenden Record ein 'F' vorkommt.

f) SE 'F1',ALL — Eingabe: Alle Auftreten suchen ab dem laufenden Record

AS C **AF1**(2)
EX P-**F1** (0)
DI 'ENDE **F1** ' (0) — Ausgabe: Alle Records, die die Zeichenfolge 'F1' enthalten.

- Notizen -

STRING ($EDIT)

Mit diesem Befehl kann eine bestimmte Zeichenfolge in einem oder mehreren EWF-Records durch eine andere Zeichenfolge gleicher oder unterschiedlicher Länge ersetzt werden.

Die STRING-Funktion beginnt immer mit dem laufenden Record.
In allen Records wird der Inhalt ab der Spalte untersucht, die mit dem COLUMN-Parameter im ASSIGN-Befehl festgelegt wurde.
(vergleiche Befehl ASSIGN).

Format:

```
STRING   'altefolge'  /WITH\ 'neuefolge' [/,ALL\]
                      \,   /               |,n  |
                                           | n  |
                                           \*d  /
```

Parameter:

WITH und Komma sind gleichbedeutend.

Die **Hochkommas** können auch durch ein beliebiges anderes Zeichen ersetzt werden. An allen 4 Stellen ist jedoch dasselbe Begrenzungszeichen zu verwenden.

z.B: ST 'F1', 'F3', ALL

kann auch eingegeben werden als:

ST XF1X, XF3X, ALL oder
ST AF1A, AF3A, ALL oder
ST &F1&, &F3&, ALL

altefolge = die zu ersetzende Zeichenfolge

neuefolge = die anstelle jeder gefundenen alten Zeichenfolge einzusetzenden Zeichen

ohne die Parameter ,n ,All und *d:
Nur der laufende Record wird untersucht. Darin wird nur die erste 'altefolge' ersetzt.

,n	=	(mit Komma vor n) maximale Anzahl sequentiell zu untersuchende Records.
n	=	(ohne Komma vor n) maximale Anzahl zu ersetzende Zeichenfolgen.
,ALL	=	Ab dem laufenden Record sind maximal alle nachfolgenden Records bis zum Dateiende zu untersuchen.
*d	=	Erst das d-te Auftreten der 'altefolge', ab dem laufenden Record wird ersetzt. z.B: *4 ersetzt erst die vierte 'altefolge', beginnend ab dem laufenden Record.

Die STRING-Funktion wird beendet, wenn soviele Records abgesucht sind, wie der n- bzw. ALL-Parameter angibt. Der laufende Record gilt als erster abzusuchender Record und zählt im n- bzw. d-Parameter mit.

Bemerkungen:

Die angegebenen Zeichenfolgen können aus allen ASCII-Zeichen bestehen.

Die 'altefolge' muss mindestens 1 Zeichen lang sein; die 'neuefolge' kann auch die Länge Null aufweisen, z.B. um Zeichen aus einem Wort zu entfernen.

Weisen die alte und die neue Zeichenfolge unterschiedliche Längen auf, werden die Daten rechts der veränderten Stelle entsprechend verschoben. Zeichen, welche dabei über die Spalte 80 hinaus geschoben würden, gehen verloren.

Im Normalfall (SUPRESS = FALSE), werden alle veränderten Records auf dem Bildschirm ausgegeben.
Wurde mit ASSIGN der Parameter SUPPRESS = TRUE festgelegt, erscheinen die veränderten Records nicht am Bildschirm. (vergleiche Texteditor-Befehl ASSIGN).

Nach einem durchgeführten STRING-Befehl erscheint die Bestätigung auf dem Bildschirm:

```
** n LINE(S) CHANGED
```

n = Anzahl der veränderten Records im EWF

Mit dem folgenden Befehl lässt im laufenden Records ganz am Anfang eine Zeichenfolge einschieben:

ST '','xxxx' fügt vorn im Record xxxx ein.

Beispiele

Annahme: COLUMN = 1, SUPRESS = FALSE (Normalfall).

Ausgangs-Zustand des EWF für alle Beispiele:

```
      ----------------------
1*    |  SET SW 1 OFF      |
2*    |  AS F1 F1 (1)      |
3*    |  AS F2 F-2 (MT)    |
4*    |  EX P-F24 (1)      |
      ----------------------
```

a) **DI *1** Eingabe
ST 'F2', 'F3', ALL

```
--------------------
|  SET SW 1 OFF    |
|  AS F1 F1 (1)    |
|  AS F3 F-2 (MT)  |
|  EX P-F34 (1)    |
--------------------
```

Zustand des EWF **nachher** Die veränderten Records (Fettdruck) werden ausgegeben.

b) **DI *2** Eingabe
ST 'F1', 'F'

```
--------------------
|  SET unverändert |
|  AS F F1 (1)     |
|  AS   unverändert|
|  EX              |
--------------------
```

Zustand des EWF
nachher: Nur die erste Folge im laufenden Record wurde verändert

c) **DI *2** Eingabe
ST 'F1', 'F', 1

SET unverändert AS **F F** (1) AS unverändert EX	Zustand des EWF **nachher:** Alle 'F1' in max. 1 Record wurden verändert.

d) **DI *2** Eingabe
ST 'F', 'D' *3

SET SW 1 OFF AS F1 F1 (1) AS **D2** F-2 (MT) EX P-F24 (1)	Zustand des EWF **nachher:** Das 3. 'F' ab Zeile 2 (einschliesslich Zeile 2) wurde ersetzt.

e) **AS CO=6** Eingabe
DI *1
ST 'F', 'FF', ALL

SET SW 1 OFF AS F1 **FF**1 (1) AS F2 **FF**-2 (MT) EX P-**FF**24 (1)	Zustand des EWF **nachher:** Alle 'F' ab Spalte 6 wurden durch 'FF' ersetzt.

f) **DI *4** Eingabe
ST '', 'P4: '

SET unverändert AS unverändert AS unverändert **P4:** EX P-F24(1)	Zustand des EWF **nachher:** 'P4: ' vorn eingefügt

TAB ($EDIT)

Dieser Befehl erlaubt das Setzen und Löschen von Tabulatoren, die bei der Einfügung neuer Records (INSERT) benützt werden können.

Formate:

TAB p [,p]...

TAB CLEAR /**p** [,p]...\
ALL /

Parameter:

- nur p ,p...	Setzen von Tabulatoren an den mit p bezeichneten Positionen im Record. Als p können Werte von 2 bis 80 eingesetzt werden. Schon vorher gesetzte Tabulatoren an andern Positionen bleiben unverändert.
- **CLEAR** p,p...	Löschen der Tabulatoren an den mit p bezeichneten Positionen. Alle andern Tabulatoren bleiben unverändert.
- **CLEAR ALL**	Löschen aller Tabulatoren.

Bemerkungen:

Um mit Tabulatoren arbeiten zu können, muss nach dem Start des Texteditors mindestens ein TAB-Befehl ausgeführt werden.

Die Anwendung der Tabulatoren ist unter dem Texteditor-Befehl INSERT beschrieben.

Beispiele:

a) Setzen eines Tabulators auf Spalte 8:
TA 8

b) Zusätzlich dazu Tabulatoren auf die Spalten 12 und 20 setzen:
TA 12,20 oder : TA 20,12

c) Den Tabulator auf Spalte 12 löschen:
TA CL 12

d) Alle gesetzten Tabulatoren löschen :
TA CL AL

30. DIE COBOL-COMPILER ($COBOL, $COBOL9, COBOL85)

Dieses Kapitel behandelt nur die Durchführung von COBOL-Compilationen. Eine ausführliche Beschreibung der COBOL-Sprache und der Programmliste finden Sie im COBOL-Manual.

30.1 UEBERSICHT

Der COBOL-Compiler baut aus einem Source-Programm auf Disc oder Magnetband ein Objectprogramm auf Disc.

Das ITX-Betriebssystem kennt 3 COBOL-Compiler:

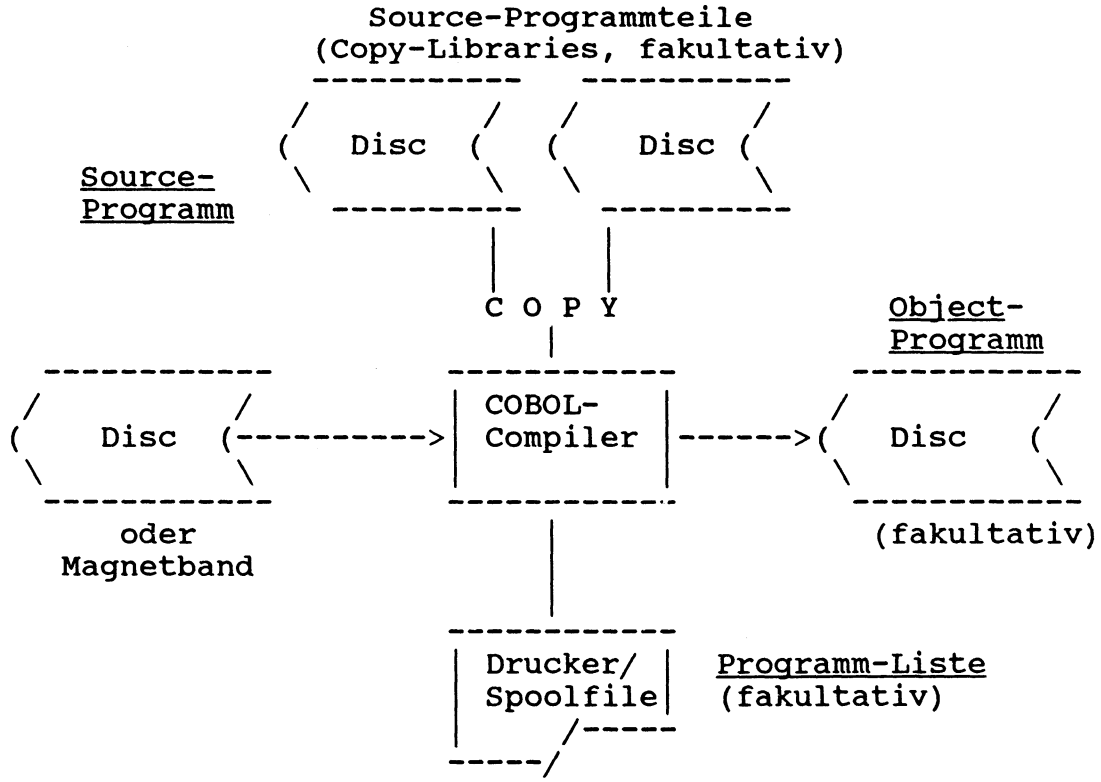
- **Der Interpretiv-Compiler ($COBOL):**
  Das Objectprogramm ist in einer Sprache, die während der Ausführung durch den COBOL-Runtime-Interpreter interpretiert wird (wie unter ITX vor Release 4.1, unter IMOS, IMX und IRX). Das Objectprogramm ist sowohl auf Disc wie im Memory sehr platzsparend.

- **Die Compiler Native-COBOL-74 ($COBOL9) und COBOL 85 ($COBOL85):**
  Das Objectprogramm ist in der Maschinensprache und braucht während der Ausführung nicht interpretiert zu werden. Native-Cobol-74-Programme sind nur ab ITX Rel.4.1, COBOL85-Programme ab ITX Rel.6.0 ausführbar. Das Objectprogramm wird grösser als interpretive Objectprogramme. Prozessorinternsive Programme laufen schneller ab.

Dieses Kapitel umfasst die Compiler $COBOL und $COBOL9.

Der COBOL85-Compiler ist im deutschen Handbuch "ITX COBOL 85" beschrieben.

Uebersicht über den Compilations-Vorgang



Das Source-Programm

Source-Programme müssen mit dem Text-Editor kompatibel sein. Dies gilt auch für Source-Programmteile, die mit COPY in die Compilation eingebaut werden.

Der Compiler führt keine Sortierung der Seiten/Zeilen-Nummern durch. Auch Zeilen ohne S/Z-Nummer (Spalte 1-6 = leer) werden richtig verarbeitet.

Copy-Library

Mit dem COBOL-Befehl COPY können Source-Zeilen aus einem oder mehreren Disc-Source-Programmen (Libraries) mitcompiliert werden. Solche Zeilen werden umgewandelt und erscheinen in der Programm-Liste. Sie werden aber nicht ins Haupt-Source-Programm eingebaut.

Das Objectprogramm

Es wird immer auf Disc gespeichet und weist den Dateityp OBJ84 auf.

Auf die Ausgabe des Objectprogrammes kann verzichtet werden, wenn nur die Programmliste mit Fehlermeldungen gewünscht wird. z.B. bei Erst-Compilationen.

Stösst der Compiler auf Fehler (Errors), welche die Umwandlung einer COBOL-Zeile verunmöglichen, wird kein Objectprogramm erstellt.

Die Programmliste

Diese Liste enthält:

- alle eingegebenen COBOL-Source-Zeilen, einschliesslich der mit COPY eingebauten Programmteile.
- Fehlermeldungen und Warnungen unter den Zeilen, welche Sprachfehler aufweisen.
- Die Adressen von Datenfeldern und Befehlen im Objectprogramm zur Erleichterung des Testens.
- Angaben über die Grösse des Objectprogramms.

Die Programmliste kann direkt auf einem Drucker ausgegeben oder auf ein Spoolfile geschrieben und später ausgedruckt werden.

Arbeits-Dateien

Der Compiler eröffnet automatisch einige temporäre Arbeitsdateien auf dem System-Disc und auf der Disc-Einheit, wo sich das Source-Programm befindet.
Die benötigte Speichergrösse ist vom Umfang und von der Struktur des zu compilierenden Programms abhängig.

Programmänderungen

Alle Programmänderungen werden mit einem Text-Editor durchgeführt. Dabei kann das alte Source-Programm durch die neue Version überschrieben oder eine neue Version erstellt werden (siehe Kapitel "SCL-Editor ED" oder "Text-Editor $EDIT").

Nach der Erstellung des neuen Source-Programms wird dieses wieder dem Compiler eingegeben. Wenn das alte Objectprogramm zuvor gelöscht wurde, darf das neue denselben Namen/Generation aufweisen, andernfalls muss der Name oder die Generation anders gewählt werden.

30.2 DURCHFUEHRUNG EINER COMPILATION

30.2.1 DATEI-ZUORDNUNG

Source-Programm:

Das Source-Programm muss mit einem ASSIGN-Befehl mit dem logischen Namen SI zugeteilt werden:

AS SI physname (n) [OW RE]

Empfehlung: Die Compilation läuft schneller ab, wenn das Source-Programm mit **OW oder RE** zugeteilt ist.

Programm-Liste:

Für die Programmliste (sofern erwünscht) ist ein ASSIGN-Befehl für einen Drucker oder ein Spoolfile erforderlich. Der logische Name muss LO lauten.

AS LO ([n,] **/LP** \ )] direkter Drucker oder
\DLP/ Auto-Spooling

oder

AS LO physname (n) **NE s AP SP** [NO] manuelles Spoolfile

Ohne diesen ASSIGN-Befehl erscheinen nur Fehlermeldungen die Zusammenfassung der Programmliste auf dem Bildschirm.

Object-Programm:

Wenn ein Objectprogramm gewünscht wird, muss der folgende ASSIGN-Befehl eingegeben werden:

AS BO programmname (n) **NE s** [AP] [PR]	neues Object-Programm
oder	
AS BO programmname (n) **OW**	altes Object-Programm überschreiben

Bemerkung:
Falls das zugeteilte Objectprogramm zu klein ist, erscheint während der Compilation die Meldung:
ASSIGNED BO FILE OF nnnn SECTORS TO SMALL, mmmm SECTORS NEEDED
Durch die Antwort (NL) wird ein genügend grosses Objectprogramm mit einer um 1 höheren Generationsnummer gebildet.

Copy-Library (Falls vorhanden):

Für jedes durch einen COPY-Befehl angesprochene Source-Programm ist ein ASSIGN-Befehl erforderlich:

AS textname physname (n)

textname = derselbe Name, der im COPY-Befehl angegeben ist.

Für den Native-COBOL-74- und den COBOL-85-Compiler gilt: COPY-Library-Files müssen nur assigned werden, wenn sie sich auf einer andern Disc-Einheit befinden wie das mit SI zugeteilte Source-Programm.

Beschleunigung der Compilation für grosse Programme:

Für grosse Source-Programme kann der folgende zusätzliche Assign zur Beschleunigung der Compilation angegeben werden:

AS COBOLWORKA physname-1 (n) **SC NE sekt**

Dieser Befehl bestimmt ein eigenes Compiler-Arbeitsfile.
Ohne diesen Assign bildet der Compiler eines oder mehrere Arbeitsfiles mit den Namen COBOLWORKA, COBOLWORKB usw. mit je 260 Sektoren auf der Disceinheit (SYS3) und verkettet diese Files untereinander.

(n) eine Disc-Einheit mit möglichst geringer Aktivität.

sekt Grösse in Sektoren. Muss gross genug sein, damit nur ein einziges Arbeitsfile benötigt wird.
Die Grösse ist abhängig von Anzahl und Länge der "Programmiererwörter" im Programm und somit von Fall zu Fall sehr verschieden.
Empfehlung: Einige tausend Sektoren.

30.2.2 START DES COMPILERS

Nach allen benötigten Dateizuordnungen wird der Compiler mit dem Befehl gestartet:

Interpretiver Compiler ($COBOL):

EX $COBOL [,D] [, option] ...

D : Alle Source-Zeilen mit einem D in der Spalte 7 (Debug-Zeilen) werden compiliert. Ohne diesen Parameter werden solche Zeilen als Kommentarzeilen behandelt.

option : frei wählbare Zusatzangaben (Options) gemäss nachfolgender Zusammenstellung von Options-Paaren.

Native-Compiler ($COBOL9):

EX $COBOL9 [,D] [, option] ...

D : Alle Source-Zeilen mit einem D in der Spalte 7 (Debug-Zeilen) werden compiliert. Ohne diesen Parameter werden solche Zeilen als Kommentarzeilen behandelt.

option : frei wählbare Zusatzangaben (Options) gemäss nachfolgender Zusammenstellung von Options-Paaren.

COBOL-85 Compiler ($COBOL85):

EX $COBOL85 [LIST] [, option] ...

LIST: Für eine Compiler-Liste ist der Parameter LIST erforderlich!
option : frei wählbare Zusatzangaben (Options) gemäss dem deutschen Handbuch "ITX COBOL 85".

Nachfolgend sind die Optionen für die beiden COBOL-74-Compiler beschrieben.

Die Beschreibung des COBOL85-Compilers finden Sie im deutschen Handbuch "ITX COBOL 85".

Bemerkung zu den Optionen der beiden COBOL 74 Compiler:

Alle Options ausser XREFONLY können statt im Befehl EX $COBOL bzw. EX $COBOL9 auch als Zeilen im COBOL-Source-Programm wie folgt angegeben werden:

Spalte 7 und 8: *$

ab Spalte 12 : Options, eine oder mehrere, bis zur Spalte 72.
Wenn verneinend, das NO dahinter, mit Leerstellte oder Komma getrennt oder auf der nächsten Zeile.
Mehrere Options durch Leerstellen oder Komma trennen.

Die Options im Programm übersteuern die entsprechenden Angaben im EX $COBOL-Befehl.

Eine Option gilt solange, bis sie durch die gegenteilige Option-Zeile verändert wird.

Beispiele:

```
Spalten-Raster: ....+....1...+....2....+....3....+
                      /$   XREF
                      *$   WARN NO
                      *$   SEQ NO  DOUBLE
```

Optionen für den Interpretiv-Compiler ($COBOL)

Option, nur wenn angegeben	stillschweigende Annahme oder mögliche Option
XREF	NO XREF
XREFONLY	
NO MERGE	MERGE
NO WARN	WARN
NO LIST	LIST
NO SEQ	SEQ
LISTOBJ	NO LISTOBJ
NO MAP	MAP
DOUBLE	SINGLE
NO CHAIN	CHAIN
NO DML	DML

Die Angaben mit NO bedeuten eine Verneinung der entsprechenden Angaben ohne NO davor.

Erklärung der Compiler-Options für den Interpretiv-Compiler:

XREF : Druck einer Cross-Reference-Liste. Diese enthält für jedes Programmierwort Name, Typ, Definitionszeile und alle Zeilen, wo dieser Namen im Programm verwendet wird. Die Cross-Reference erscheint zusätzlich zur Compilation.

XREFONLY : Nur Source-Zeilen und Cross-Reference-Liste drucken, ohne zu compilieren.

MERGE : Fehlermeldungen und Warnungen werden unmittelbar nach der betreffenden COBOL-Zeile ausgegeben.
Mit NO MERGE erscheinen alle diese Meldungen am Ende der Programm-Liste.

Erklärung der Compiler-Options für den Interpretiv-Compiler (Fortsetzung):

WARN	:	Warnungen ausgeben. Dies sind Compiler-Meldungen, welche die Compilation nicht behindern. Mit NO WARN werden die Warnungen nicht ausgegeben.
LIST	:	Ausgeben aller Source-Zeilen. Mit NO LIST werden die Source-Zeilen nicht ausgegeben.
SEQ	:	Eine Warnung ausgeben, wo im Source-Programm absteigende Seiten/Zeilen-Nummern auftreten.
LISTOBJ	:	Auch das Object-Coding ausgeben.
MAP	:	Ausgabe des Daten-Bereich-Verzeichnisses, das für den Debug-Modus gebraucht wird.
NO MAP	:	Gibt dieses Verzeichnis nicht aus.
DOUBLE	:	Nur jede zweite Zeile bedrucken. Mit Raum für Notizen zwischen den Zeilen.
SINGLE	:	Jede Zeile bedrucken.
CHAIN	:	Nach jeder Fehlermeldung/Warnung wird angegeben, auf welcher Seite sich der letzte vorangehende Fehler befindet.
DML	:	Die Datenbank-Befehle für DBS-IRX bzw. ITX-DBS werden berücksichtigt.
NO DML	:	Mit NO DML gelten solche Datenbank-Befehle als Fehler. Die für Datenbank-Befehle reservierten COBOL-Wörter können frei verwendet werden. Der Compiler ist ausserdem schneller.

Optionen für den NATIVE COBOL 74 Compiler ($COBOL9)

Die Angaben mit NO bedeuten eine Verneinung der entsprechenden Angaben ohne NO davor.

BOUNDS : Beim Zugriff auf eine variable Tabelle (OCCURS... DEPENDING) wird getestet, ob der Subscript bzw. Index einen erlaubten Wert aufweist.
Ohne Angabe gilt: NO BOUNDS

CHAIN : Nach jeder Fehlermeldung/Warnung wird angegeben, auf welcher Seite sich der letzte vorangehende Fehler befindet.
Falls nicht erwünscht: NO CHAIN angeben.

NO COPY : Spart Compilerzeit, wenn das Source-Programm keine COPY-Befehle enthält.

Ohne Angabe gilt: COPY. COPY ist erforderlich, wenn das Source-Programm COPY-Befehle ab einer Source-Library enthält.

NO CUE : Die Namen der durchlaufenen Compiler-Phasen werden am Bildschirm nicht ausgegeben.
Ohne Angabe gilt: CUE.

DYNAMIC : Das Programm kann "dynamische" CALL's enthalten, welche das CALL-Modul vom Disc laden.
Wird NO DYNAMIC angegeben, müssen Hauptprogramm und CALL-Moduls vor der Ausführung zusammengelinkt werden mit dem Utility $LINK.
Ohne Angabe gilt: DYNAMIC.

HEAP=nK : Grösse des verwendbaren Memory-Heap-Bereiches für die Compilation in KBytes.
Als n kann 1 bis 32 angegeben werden.
Dieser Parameter ist notwendig, wenn der Compiler mit der Meldung OUT OT MEMORY abgebrochen wird.
Ohne Angabe gilt: HEAP=4K.

Optionen für den NATIVE COBOL 74 Compiler ($COBOL9)
Fortsetzung:

ICMP : Bei der Verarbeitung numerischer Felder gelten die Regeln des Compilers $COBOL: Es werden nur die rechten 4 Bits jedes Byte verarbeitet, die linken 4 Bits werden auf hex. 3 gesetzt.
Empfehlenswert bei der Native-Compilation bestehender Programme, die früher interpretiv compiliert und getestet wurden.

LIST : Ausgeben aller Source-Zeilen. Mit NO LIST werden die Source-Zeilen nicht ausgegeben.
Ohne Angabe gilt: LIST.

LISTOBJ : Auch das Object-Coding ausgeben.
Ohne Angabe gilt: NO LISTOBJ.

LP=n : Angabe des direkten Druckers (im Sinne von DLP) für die Compilerliste, falls kein AS LO... angegeben wurde.

LP : Nur für den direkten Drukc (n DLP) der Compiler-Liste: Der Drucker wird vom Bediener verlangt, sobald ihn der Compiler benötigt. Für solche Compilationen ist kein AS LO... erforderlich.

MAP : Der Compiler gibt eine Liste der Datenbereiche mit Speicheradressen des Programms aus.
Ohne Angabe gilt: NO MAP.

MERGE : Fehlermeldungen und Warnungen werden unmittelbar nach der betreffenden COBOL-Zeile ausgegeben.
Mit NO MERGE erscheinen alle diese Meldungen am Ende der Programm-Liste.
Ohne Angabe gilt: MERGE.

MIGRATION : Der COBOL-Befehl DISPLAY Datenname-1 Datenname-2 ... gibt die Datennamen auf jeweils eine neue Zeile aus.
NO MIGRATION gibt die Datennamen auf dieselbe Zeile aus.
Ohne Angabe gilt: NO MIGRATION.

Optionen für den NATIVE COBOL 74 Compiler ($COBOL9)
Fortsetzung:

M9300 oder M9400:
Optimalisierung des Object-Codings auf den Prozessor 9300 oder 9400.
Ohne Angabe gilt: M9400.

SEGSIZE=nK : Zuordnung von Firmware-Segmenten in KBYTES. als n kann 1 bis 32 angegeben werden.
Eine solche Angabe ist notwendig, falls der Compiler mit der Meldung STORx SEGMENTS EXHAUSTED abgebrochen wird.
Ohne Angabe gilt: SEGSIZE=4K.

SEQ : Eine Warnung ausgeben, wo im Source-Programm absteigende Seiten/Zeilen-Nummern auftreten.
NO SEQ gibt keine solchen Warnungen aus.
Ohne Angabe gilt: SEQ.

SINGLE oder DOUBLE:
Abstand der Zeilen auf der Programmliste:
SINGLE oder keine Angabe: einfacher Abstand,
DOUBLE: Doppelter Abstand (1 Leerzeile zwischen zwei Zeilen).

SIZE : Programmabbruch, wenn bei einer arithmetische Anweisung ohne SIZE ERROR-Klausel das Ergebnisfeld überläuft.
NO SIZE macht diesen Test nicht.
Ohne Angabe gilt: NO SIZE.

SYMTBL : Im Object-Programm sollen auch die Tabellen generiert werden, die für die Testhilfe "Symbolic Debug" erforderlich sind. Das Object-Programm wird dabei rund doppelt so gross.

TRUNC : In einem binären Empfangsfeld wird gemäss Dezimalstellen-Angabe im PICTURE angeglichen und überzählige Stellen abgeschnitten.
NO TRUNC den gemäss Feldgrösse maximalen Wert.
Ohne Angabe gilt TRUNC.

Optionen für den NATIVE COBOL 74 Compiler ($COBOL9)
Fortsetzung:

WARN : Warnungen ausgeben. Dies sind Compiler-Meldungen, welche die Compilation nicht behindern.
Mit NO WARN werden die Warnungen nicht ausgegeben.
Ohne Angabe gilt: WARN

XREF : Druck einer Cross-Reference-Liste. Diese enthält für jedes Programmierwort Name, Typ, Definitionszeile und alle Zeilen, wo dieser Namen im Programm verwendet wird. Die Cross-Reference erscheint zusätzlich zur Compilation.
Ohne Angabe gilt NO XREF.

XREFONLY : Nur Source-Zeilen und Cross-Reference-Liste drucken, ohne zu compilieren.

XSPACE : Beim Vergleich eines Feldes mit Vorzeichen mit einem alphanumerischen Feld wird das Vorzeichen ignoriert.
NO XSPACE vergleicht das Vorzeichen mit.
Ohne Angabe gilt XSPACE.

30.3 STRUKTUR DES OBJECT-PROGRAMMS

Ein COBOL-Objectprogramm besteht aus lokalen und globalen Segmenten.

Die als "global" bezeichneten Segmente werden nur einmal im System gespeichert, wenn dasselbe Programm gleichzeitig in mehreren Prozessen ausgeführt wird (GLOBAL BYTE SIZE).

Die als "local" bezeichneten Segmente werden für jeden Prozess, welcher das Programm ausführt, einzeln gespeichert (LOCAL BYTE SIZE).

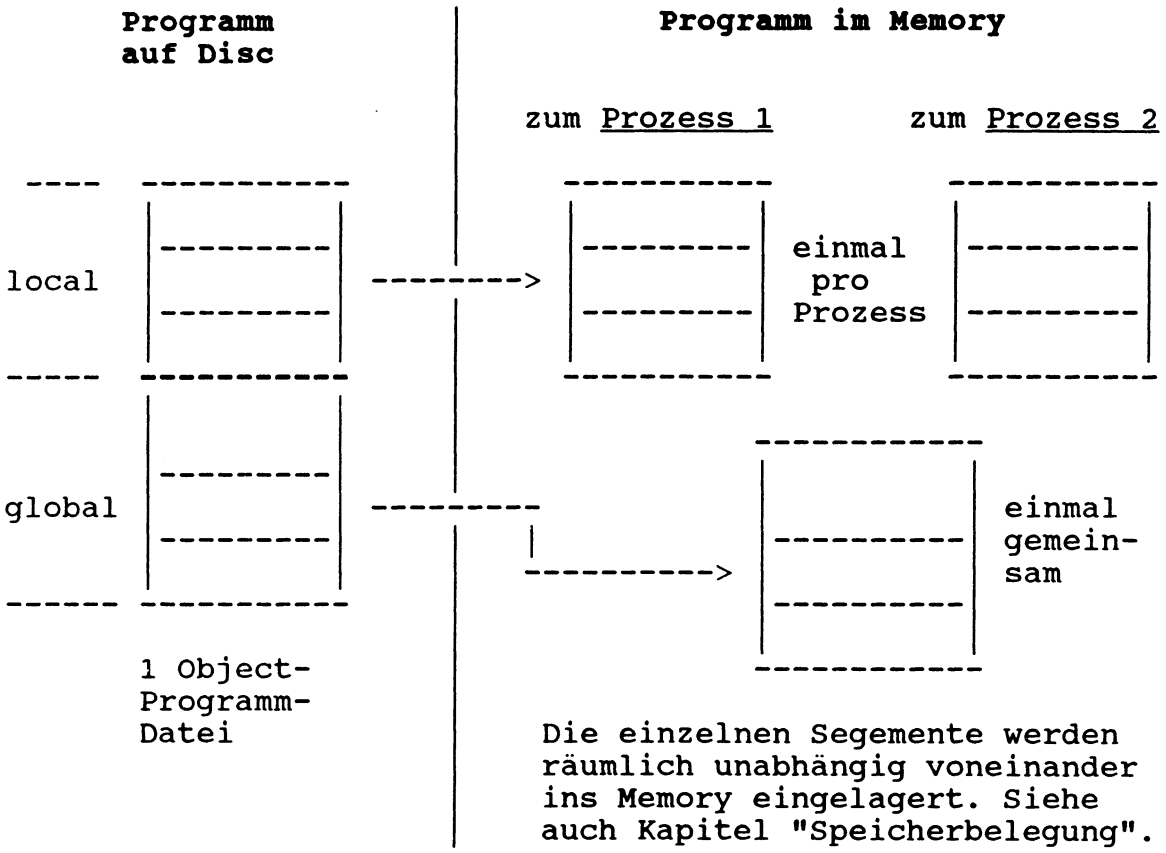
Die Local Segemente sind:

- Steuerblock für das Programm (RIA, Runtime interface area)
- File- und Working-Storage Section (Local Data)
- Speicherblock für temporäre Informationen (Temporary storage)
- Der Ueberlagerungsbereich, gross genug für die grösste Ueberlagerung.

Die Global-Segmente sind:

- Befehle und Literale der Procedure Division (Resident coding und Resident literals)
- Die Global Section (Global Data).
- Das Ueberlagerungs-Verzeichnis

Speicherung eines Programms, das zweimal im System geladen ist:



Jedes mit CALL von andern Programmen aufgerufenes (Unter)-Programm wird genau gleich aufgeteilt:

Seine "local"-Teile werden einmal pro aufrufenden Prozess, seine "global"-Teile nur einmal im ganzen gespeichert.

- Notizen -

31. SPEICHER-FORMATE AUF MAGNETISCHEN DATEN-TRAEGERN

31.1 DISC

31.1.1 PLATTEN-EINTEILUNG

DRIVES

Die Platten-Einheiten, welche durch einen einzigen Zugriffsmechanismus bedient werden, bilden einen Drive oder eine physische Disc-Einheit. Ein Drive wird immer als ganzes ein- und ausgeschaltet (sofern er überhaupt einzeln zuschaltbar ist).

In der Regel besteht ein Drive aus mehreren (logischen) Disc-Einheiten.

DISC-EINTEILUNG DER VERSCHIEDENEN DISC-TYPEN

Disc-Typ	Total Megabytes	Total Anz. Sektoren	Anzahl Grund-Units (Fix v. Hersteller)	Sektoren pro Grund-Unit
Fix-Discs:				
6514	66,5	129'856	4	32'464
6516	138,5	266'310	10	26'631
6518	320	610'540	20	30'527
6528	135	261'120	10	26'112
6542	270	540'672	16	33'792
6543	400	819'072	24	33'792
6545	304	595'840	16	37'142
6546	410	798'600	25	31'944
wechselbare Discs:				
6524	67,5	130'560	5	26'112
WORM-Disc (Write Once Read Many, Archivierungs-Disc):				
6091-3001	653	1'272'392 2 Seiten. Nachdem die eine Seite beschrieben ist, muss der Disc gewendet werden.	2	636'346

Die Gruppierungs-Möglichkeiten der Grund-Units zu logischen Einheiten finden Sie im Kapitel "Disc-Initialisierung, $DINT" im ersten Band.

Einteilung einer logischen Disc-Einheit:

Segment	Von/bis Sektor	Anzahl Sektoren
Volume Header	0	1
reserviert	1	1
Ersatzspuren-Verzeichnis	2	1
Disc-Directory 1)	3 - max. 412	10 - 410
Bereich für System und Benützer-Dateien	verbleibende Anzahl Sektoren	
Ersatzspuren 2)	je nach Disc-Typ	

1) Grösse des Disc-Directory:
10 bis 410 Sektoren, abhängig von der erwarteten Anzahl Files.

2) Grösse des Ersatzbereiches:
Je nach Disc-Typ, 84 bis 5700 Sektoren. In der Regel wird die "default"-Grösse beim Initialisieren verwendet.

WORM-Disc (Archivierungs-Disc):

Jeder Sektor dieses optischen Disc kann nur einmal beschrieben werden. Er wird initialisiert geliefert. Sein Aufbau weicht vom Aufbau der magnetischen Disc-Typen ab: Directory-Grösse auf jeder Seite 3280 Sektoren für je 3280 Dateien.

VOLUME HEADER UND ERSSATZSPUREN-VERZEICHNIS

Die genaue Beschreibung dieser Bereiche finden Sie im amerikanischen Original-Reference-Manual, Kapitel "Disk Formats".

DISC-DIRECTORY-AUFBAU

Die Records Disc-Directory haben eine fixe Länge von 64 Bytes. 8 Dateien (Sektor
1. Record:
Disc-Directory Section Header, 64 Bytes.
Die genaue Beschreibung finden Sie im amerikanischen Original-Reference-Manual, Kapitel "Disk Formats".
Weitere Records:
Datei-Eintragungen, 1 Record pro Datei.

Einteilung der Datei-Eintragungen im Disc-Directory:

Position im Record relativ 0	Länge (Bytes)	Feldbezeichnung	Typ	Bemerkungen
0	1	Entry-Type	binär	Hex 00=gültige, Hex FF=gelöschte Datei Hex 5E=Ende des Directory oder leerer Sektor, übrige Werte=System-Elemete
1	1	Hold-Type	binär	Hex 01=scratch 03=permanent
2	1	reserviert		immer Hex 00
3	1	Sperr-Flag	binär	gesperrt = 01 löschbar = 00
4	4	Start-Sektor	binär	
8	4	End-Sektor	binär	
12	17	Dateiname	ASCII	
29	2	Section-Nr.	binär	
31	1	Section-Indikatior		Hex 01 = letzte Section Hex 00 = nicht letzte Section
32	16	Erstell-Datum Form:JJJJMMTTSSMISEHH JJJJ=Jahr,MM=Monat,TT=Tag St=Std. MI-Min.,SE=Sek. HH=Hundertst.Sek.	ASCII	

Einteilung der Datei-Eintragungen im Disc-Directory (Fortsetzung):

Position im Record relativ 0	Länge (Bytes)	Feldbezeichnung	Typ	Bemerkungen
48	2	Generation	binär	
50	2	Version	binär	immer Hex 0000
52	6	Verfall-Datum	ASCII	immer 999999
58	1	Multi-Volume-Indikator	ASCII	0=ganzes File auf derselben Einheit 1=Multi-Volume-File
59	5	reserviert		immer alles Hex00

Die Directory-Angaben, die hier nicht zu finden sind, befinden sich im Start-Sektor der betreffenden Datei, dem sogenannten Section Label.

Section Label
(Directory-Angaben im ersten Sektor der Datei):

Der erste Sektor jeder Datei-Section ist der Section Label.
Die Daten beginnen erst ab dem 2. Sektor jeder Section.

Position im Record rel.0	Länge (Bytes)	Feldbezeichnung	Typ	Bemerkungen
0	4	fixer Wert		immer Hex02000000
4	6	Label-Identifikation	ASCII	immer SXNLBL
10	2	Section-Nummer	binär	immer Hex 0001
12	17	Dateiname	ASCII	
29	16	Erstellungs-Datum und Zeit	ASCII	Form:JJJJMMTTSTMISEHH(vergl.Disc-Directory)
45	2	Generation	binär	
47	2	Version	binär	immer Hex 0000
49	1	Entry-Type	binär	Hex 00 =gültige Datei (vergleiche Disc-Directory)
50	1	Datei-Typ	binär	0 = Daten 4 = COBOL-Object-Programm 9 = manuelles Spoolfile
51	1	fixer Wert		immer Hex 00
52	4	Blocklänge	binär	
56	4	max.Recordlänge	binär	
60	4	min.Recordlänge	binär	
64	1	Record-Format	binär	0 = fix 1 = varibel

Section Label, Fortsetzung
(Directory-Angaben im ersten Sektor der Datei):

Position im Record rel. 0	Länge (Bytes)	Feldbezeichnung	Typ	Bemerkungen
65	1	Datei-Organisation	binär	0 = sequential 1 = relative 2 = indexed
66	12	fixer Wert	binär	immer Hex null
78	1	Privat-Anzeige	binär	0 = sharable 1 = private
79	1	fixer Wert	binär	immer Hex 00
80	4	Start-Data, relativ zum Start-Sektor	binär	immer 1
84	4	End-Data, relativ zum Start-Sektor +1	binär	
88	4	End-Sektor, relativ zum Start-Sektor +1	binär	
92	1	Daten-Code	binär	immer 0 für ASCII-Code
93	16	unbenützt	ASCII	immer Hex 00
109	6	Verfall-Datum	ASCII	Form: JJMMTT immer 999999
115	1	fixer Wert	binär	immer Hex 00
116	1	File-Status	binär	0 = abgeschlossen 2 = eröffnet 5 = gelöscht

Section Label, Fortsetzung
(Directory-Angaben im ersten Sektor der Datei):

Position im Record rel. 0	Länge (Bytes)	Feldbezeichnung	Typ	Bemerkungen
117	1	AP-Anzeige	binär	1 = ohne AP 0 = mit AP zuget.
118	1	Section-Anzeige	binär	0 = es gibt weitere Folge-Sections 1 = dies ist die letzte Section
119	1	fixer Wert	binär	0
120	30	fixer Wert	ASCII	immer SPACES
150	13	System-Code, File-Erstellung	ASCII	z.B.NCR-ITX-01-00
163	13	System-Code, aktuelle Eröffnung	ASCII	dito.
176	13	System-Code, letzte Erffnung	ASCII	dito.
189	3	System, wofür das File gilt (nur Objectprogramme)	ASCII	IRX/ITX = IRX bzw. ITX IMS = IMOS 3/5, IMX
192	16	Eröffnung bzw. Abschluss-Datum und -Zeit	ASCII	Form : siehe Erstellungs-Datum
208	4	fixer Wert	binär	immer Hex00000000
212	4	Byte Offset des nächsten freien Records	binär	
216	38	fixer Wert	binär	immer Hex 00
258	256	fixer Wert	ASCII	immer SPACES

Bemerkungen zum Disc-Directory

Gelöschte Dateien:

gelöschte Dateien werden wie folgt gekennzeichnet.

- im Disc-Directory erhält der ENTRY TYPE den Wert Hex FF
- im Section Header bleibt der ENTRY TYPE = Hex 00, doch wird das Feld FILE STATUS auf 5 gesetzt.

Reaktivieren gelöschter Dateien:

Solange keine andere Datei den Platz der gelöschten einnimmt und das Disc-Directory nicht komprimiert wird (MOUNT mit CO), lässt sich eine gelöschte Datei reaktivieren:

- Im Disc-Directory in der vordersten Eintragung des Files mit dem Befehl FIX den ENTRY TYPE auf HEX 00 zurücksetzen:
  - Mit dem Sektor-Dump das Disc-Directory ausgeben.
  - Von vorn her die erste Eintragung mit dem Namen der gelöschten Datei suchen.
  - Der Entry-Type steht 11 Bytes vor dem Physnamen (vgl. vorangehende Seiten!)..
- Die Datei mit CHE physname (n) bereinigen.

31.1.2 DISC-PLATZBEDARF FUER DAS BETRIEBSSYSTEM

Damit alle System-Elemente und System-Arbeitsbereiche auf dem System-Disc genügend Platz finden, empfehlen wir, die folgende Anzahl logische Units (Grund-Units) für das System (SYS1, SYS2 und SYS3 zusammen) zu reservieren.
Der Systemdisc SYS1 muss sich immer ganz am Anfang eines physischen Drive befinden!

Die angegebenen Grössen reichen für die gesamte Palette der ITX-Systemsoftware und genügen für die maximale angegeben Memory-Grösse (Swapfile!).

System/ Modell	Disc-Typ/ MB pro phys. Drive	Memorygrösse in MB	Anzahl log. Grundunits für das System
10000/35	6516/135MB	bis 4 MB	6
10000/55	6516/135MB	bis 8 MB	8
		9 bis 16 MB	10 (1 Drive)
10000/65	6528/135MB	bis 16 MB	10 (1 Drive)
	6545/300MB	17 bis 32 MB	16 (1 Drive)
	6518/300MB	bis 16 MB	10
		17 bis 32 MB	16
10000/75	6528/135MB	bis 16 MB	10 (1 Drive)
	6545/300MB	17 bis 32 MB	16 (1 Drive)
	6518/300MB	bis 16 MB	10
		17 bis 32 MB	16

31.2 MAGNETBAND

31.2.1 BAND-AUFBAU

a) Mit Kennsätzen, ANSI oder NON-STANDARD:

Solche Bänder enthalten die folgenden Kennsätze (Labels):
Jeder Kennsatz ist 80 Bytes lang und bildet einen eigenen Block. Alle seine Felder sind ungepackt.

VOL: Spulen-Vorsatz, 1 pro Spule, d rch Initialisieren ($MINT) geschrieben.

HDR1, HDR2 und eventuell HDR3 und HDR4:
Datei-Vorsätze, bei der Datei-Eröffnung geschrieben.

EOF1, EOF2 und eventuell EOF3 und EOF4:
Datei-Nachsätze, beim Datei-Abschluss geschrieben.

EOV1, EOV2 und eventuell EOV3 und EOV4:
Spulen-Nachsätze, wenn eine Datei über mehrere Spulen reicht, am Ende jeder Spule, welche nicht die letzte ist.

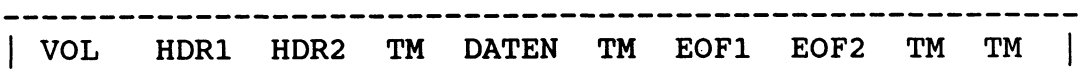
Tape-Mark (TM):
Bandmarken zwischen Kennsätzen und Daten.

HDR3 und 4, EOF3 und 4 sowie EOV3 und 4 erscheinen nur auf Non-Standard-Bändern.

Beispiele von Band-Strukturen

ANSI-Standard

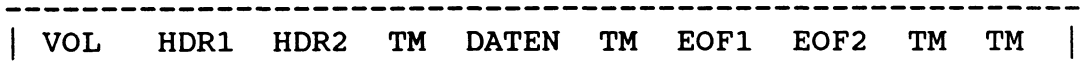
- 1 Datei auf einer Spule:



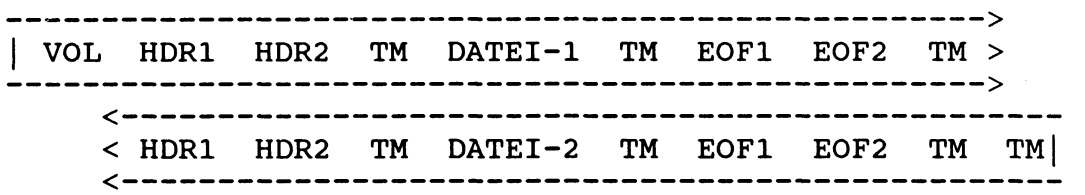
- 1 Datei auf 2 Spulen:

1. Spule

2. Spule (Fortsetzung)



- 2 Dateien auf 1 Spule:



Folgende Labels werden von ITX unterstützt:

- VOLUME HEADER (VOL1)
- FILE HEADER 1 bis 4 (HDR1 bis HDR4)
- END OF VOLUME 1 bis 4 (EOV1 bis EOV4)
- END OF FILE 1 bis 4 (EOF1 bis EOF4)

Nicht unterstützt sind Benützer-Labels.
Wird ein File mit Open Output oder Extend verarbeitet, ist zu beachten, dass beim Close die IRX/ITX-Schluss-Labels (Trailer Labels) geschrieben werden. Ist das File nicht unter IRX oder ITX erstellt worden, werden allfällig bereits vorhandene Labels ignoriert und die IRX/ITX-Labels dahinter gesetzt.

b) Magnetbänder ohne Kennsätze

Solche Bänder bestehen nur aus Daten-Blöcken, am Bandende werden Tape-Marks gesetzt.
Jeder Block muss als 1 Record behandelt werden.
Eine Datei kann höchstens eine Spule umfassen und eine Spule nur 1 Datei enthalten.

Bänder mit dieser Struktur sind mit dem MOUNT-BEFEHL mit dem Parameter AS (falls ASCII-Code) oder EB (Falls EBCDIC-Code) zu mounten. Ihr Inhalt kann mit DUMP ausgegeben werden. Andere Verarbeitungen sind nur mit Programmen möglich.

31.2.2 AUFBAU DER BAND-KENNSAETZE

Die genaue Beschreibung dieser Bereiche finden Sie im amerikanischen Original-Reference-Manual, Kapitel "Tape Formats".

31.2.3 KURZE BLOECKE

Blöcke unter 18 Bytes Länge werden auf dem Band automatisch mit Hex 5E auf die Länge von 18 Bytes aufgefüllt.

Durch andere Systeme erstellte Bänder können mit IRX/ITX nur gelesen werden, wenn Blöcke von weniger als 16 Bytes Länge am Ende mindestens 3 Bytes mit Hex 5E enthalten. Blöcke von 16 oder 17 Bytes müssen mit Hex 5E auf 18 Bytes aufgefüllt sein.

31.3 STREAMER-BAND UND HELICAL SCAN TAPE

Der Aufbau dieser Bandarten ist spezifisch für NCR-ITX. Er ist deshalb in diesem Manual nicht beschrieben.

Die Beschreibung der Streamer-Labels finden Sie im amerikanischen Original-Reference-Manual, Kapitel "Tape Formats".

- Notizen -

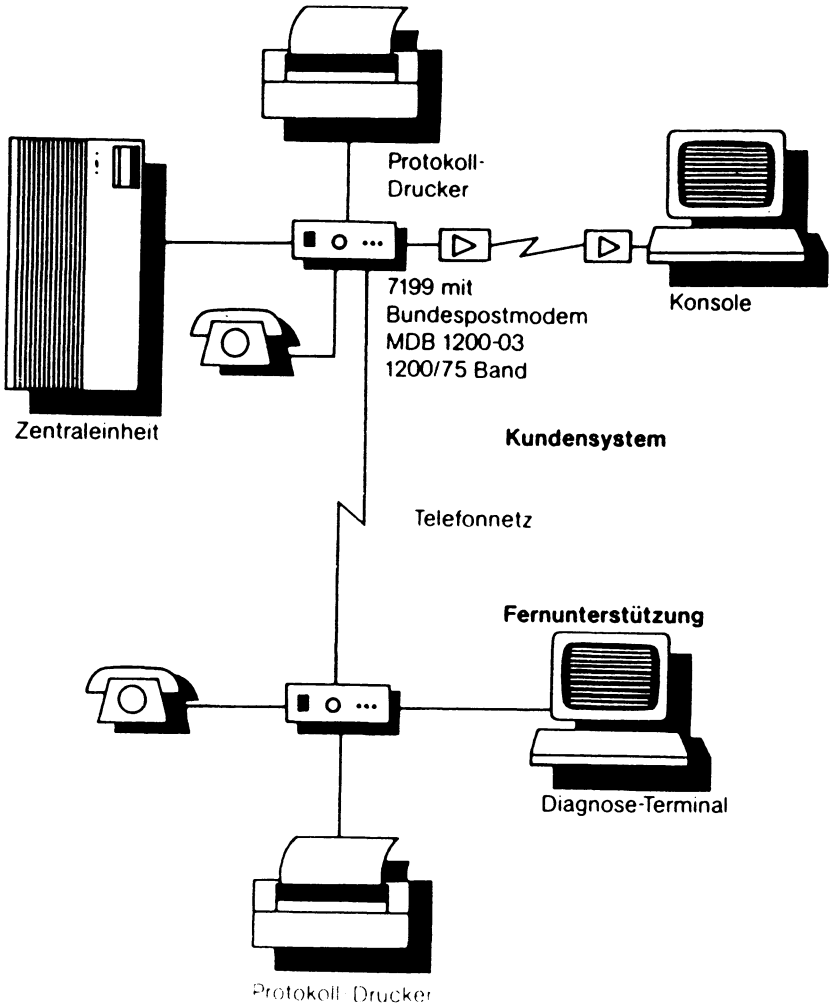
32. NCR-FERN-UNTERSTUETZUNG

Erhöhte Systemverfügbarkeit durch rasche Störungsbehebung.

Die Fernunterstützung ermöglicht es NCR Soft- und Hardware Spezialisten innerhalb weniger Minuten **"vor Ort"** tätig zu werden.

Die Diagnose einer Systemstörung kann entweder verbal am Telefon oder über den FU-Processor (siehe Abbildung) erfolgen.

Typischer FU-Anschluß mit 7199:



"Im Fall des Falles"

Um einen Systemausfall innerhalb kürzester Frist zu beheben, informieren Sie bitte möglichst rasch ihre NCR Kundendienst-Leitstelle oder den Zentralen Software Service in Augsburg.

Sie werden baldmöglichst von dem zuständigen Spezialisten angerufen der mit Ihnen die Ferndiagnose durchführt.

Ablauf der Fernunterstützung

Der Soft- oder Hardwarespezialist wird das Problem zuerst mit Ihnen am Telefon besprechen. Ist zur weiteren Diagnose eine Datenverbindung mit Ihrem EDV-System nötig, dann müssen Sie Ihren FU-Processor aktivieren und nach Absprache die Datentaste an Ihrem Telefon drücken.

Der Spezialist wird nun über seinen und Ihren FU-Processor eine Datenverbindung aufbauen und ist nun in der Lage, auf Ihrem EDV-System zu arbeiten.

Datenschutz

Sämtliche Ein- und Ausgaben erscheinen gleichzeitig auf dem Kundenterminal und auf dem Diagnose-Bildschirm bei NCR. Sie können jederzeit verfolgen, was der Spezialist gerade macht.

Nur durch aktive kundenseitige Mitarbeit kann die Verbindung zum Kundensystem aufgebaut werden.

Ein Abbruch der Datenverbindung durch den Kunden ist jederzeit möglich.

Wenn Sie noch weitere Fragen zur Fernunterstützung haben, dann wenden Sie sich bitte an Ihre NCR Kundendienst-Leitstelle oder an die Fernunterstützung der NCR in Augsburg.

ANHANG

ASCII-CODE für ITX

ASCII-Code

B8-B5 \ B4-B1		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC 1	DC 2	DC 3	DC 4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	¤	!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	¯	DEL

Bemerkung:
Im deutschen Sprachraum sind die Zeichen
[, \ und] (hex. 5B, 5C und 5D) durch die Zeichen
Ä, Ö und Ü ersetzt.

ASCII-CODE für PC

Dez.	Hex	Zeichen	Dez.	Hex	Zeichen	Dez.	Hex	Zeichen	Dez.	Hex	Zeichen
0	00		31	1F	▼	62	3E	>	93	5D	]
1	01	☺	32	20		63	3F	?	94	5E	^
2	02	●	33	21	!	64	40	@	95	5F	_
3	03	♥	34	22	"	65	41	A	96	60	`
4	04	♦	35	23	#	66	42	B	97	61	a
5	05	♣	36	24	$	67	43	C	98	62	b
6	06	♠	37	25	%	68	44	D	99	63	c
7	07	•	38	26	&	69	45	E	100	64	d
8	08	◘	39	27	'	70	46	F	101	65	e
9	09	○	40	28	(	71	47	G	102	66	f
10	0A	◙	41	29	)	72	48	H	103	67	g
11	0B	♂	42	2A	*	73	49	I	104	68	h
12	0C	♀	43	2B	+	74	4A	J	105	69	i
13	0D	♪	44	2C	,	75	4B	K	106	6A	j
14	0E	♫	45	2D	-	76	4C	L	107	6B	k
15	0F	☼	46	2E	.	77	4D	M	108	6C	l
16	10	►	47	2F	/	78	4E	N	109	6D	m
17	11	◄	48	30	0	79	4F	O	110	6E	n
18	12	↕	49	31	1	80	50	P	111	6F	o
19	13	‼	50	32	2	81	51	Q	112	70	p
20	14	¶	51	33	3	82	52	R	113	71	q
21	15	§	52	34	4	83	53	S	114	72	r
22	16	▬	53	35	5	84	54	T	115	73	s
23	17	↨	54	36	6	85	55	U	116	74	t
24	18	↑	55	37	7	86	56	V	117	75	u
25	19	↓	56	38	8	87	57	W	118	76	v
26	1A	→	57	39	9	88	58	X	119	77	w
27	1B	←	58	3A	:	89	59	Y	120	78	x
28	1C	∟	59	3B	;	90	5A	Z	121	79	y
29	1D	↔	60	3C	<	91	5B	[	122	7A	z
30	1E	▲	61	3D	=	92	5C	\	123	7B	{

Dez.	Hex	Zeichen	Dez.	Hex	Zeichen	Dez.	Hex	Zeichen	Dez.	Hex	Zeichen
124	7C	|	157	9D	¥	190	BE	╛	223	DF	▀
125	7D	}	158	9E	₧	191	BF	┐	224	E0	α
126	7E	~	159	9F	ƒ	192	C0	└	225	E1	ß
127	7F	⌂	160	A0	á	193	C1	┴	226	E2	Γ
128	80	Ç	161	A1	í	194	C2	┬	227	E3	π
129	81	ü	162	A2	ó	195	C3	├	228	E4	Σ
130	82	é	163	A3	ú	196	C4	─	229	E5	σ
131	83	â	164	A4	ñ	197	C5	┼	230	E6	µ
132	84	ä	165	A5	Ñ	198	C6	╞	231	E7	τ
133	85	à	166	A6	ª	199	C7	╟	232	E8	Φ
134	86	å	167	A7	º	200	C8	╚	233	E9	Θ
135	87	ç	168	A8	¿	201	C9	╔	234	EA	Ω
136	88	ê	169	A9	⌐	202	CA	╩	235	EB	δ
137	89	ë	170	AA	¬	203	CB	╦	236	EC	∞
138	8A	è	171	AB	½	204	CC	╠	237	ED	φ
139	8B	ï	172	AC	¼	205	CD	═	238	EE	ε
140	8C	î	173	AD	¡	206	CE	╬	239	EF	∩
141	8D	ì	174	AE	«	207	CF	╧	240	F0	≡
142	8E	Ä	175	AF	»	208	D0	╨	241	F1	±
143	8F	Å	176	B0	░	209	D1	╤	242	F2	≥
144	90	É	177	B1	▒	210	D2	╥	243	F3	≤
145	91	æ	178	B2	▓	211	D3	╙	244	F4	⌠
146	92	Æ	179	B3	│	212	D4	╘	245	F5	⌡
147	93	ô	180	B4	┤	213	D5	╒	246	F6	÷
148	94	ö	181	B5	╡	214	D6	╓	247	F7	≈
149	95	ò	182	B6	╢	215	D7	╫	248	F8	°
150	96	û	183	B7	╖	216	D8	╪	249	F9	∙
151	97	ù	184	B8	╕	217	D9	┘	250	FA	·
152	98	ÿ	185	B9	╣	218	DA	┌	251	FB	√
153	99	Ö	186	BA	║	219	DB	▄	252	FC	ⁿ
154	9A	Ü	187	BB	╗	220	DC	▌	253	FD	²
155	9B	¢	188	BC	╝	221	DD	▐	254	FE	■
156	9C	£	189	BD	╜	222	DE	▀	255	FF	

UMRECHNUNG DEZIMAL-BINAER-HEX

BINAER	HEX	5	4	3	2	1
0000	0	0	0	0	0	0
0001	1	65 536	4 096	256	16	1
0010	2	131 072	8 192	512	32	2
0011	3	196 608	12 288	768	48	3
0100	4	262 144	16 384	1 024	64	4
0101	5	327 680	20 480	1 280	80	5
0110	6	393 216	24 576	1 536	96	6
0111	7	458 752	28 672	1 792	112	7
1000	8	524 288	32 768	2 048	128	8
1001	9	589 824	36 864	2 304	144	9
1010	A	655 360	40 960	2 560	160	10
1011	B	720 896	45 056	2 816	176	11
1100	C	786 432	49 152	3 072	192	12
1101	D	851 968	53 248	3 328	208	13
1110	E	917 504	57 344	3 584	224	14
1111	F	983 040	61 440	3 840	240	15

EBCDIC - CODE

Dieser Code wird auf NCR-I-Systemen nur auf Magnetband verwendet.

Bits	1-4	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
5-8	Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	PF	HT	LC	DEL	GE	RLF	SMM	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	TM	RES	NL	BS	IL	CAN	EM	CC	CU1	IFS	IGS	IRS	IUS
0010	2	DS	SOS	FS		BYP	LF	ETB	ESC			SM	CU2		ENQ	ACK	BEL
0011	3			SYN		PN	RS	UC	EOT				CU3	DC4	NAK		SUB
0100	4	∅										¢	.	<	(	+	|
0101	5	&										!	$	*	)	;	¬
0110	6	–	/									¦	,	%	_	>	?
0111	7											:	#	@	'	=	"
1000	8		a	b	c	d	e	f	g	h	i						
1001	9		j	k	l	m	n	o	p	q	r						
1010	A		~	s	t	u	v	w	x	y	z						
1011	B																
1100	C	{	A	B	C	D	E	F	G	H	I						
1101	D	}	J	K	L	M	N	O	P	Q	R						
1110	E	\		S	T	U	V	W	X	Y	Z						
1111	F	0	1	2	3	4	5	6	7	8	9	LVM					EO

FILE-STATUS - WERTE (COBOL 74)

00 = Zugriff erfolgreich abgeschlossen

02 = Erfolgreicher Zugriff mit erlaubtem Duplicate Key

0B = Tape-Mark nach Band-Ende
(nur auf Magnetband)

0E = Tape-Mark gefunden (nur auf Magnetband)

0F = Auf Disc: Warnung: Datei wird voll.
Normalfall: Datei zu mindestens 95 % gefüllt. Im Sysgen kann auch eine andere Limite definiert werden.

auf Magnetband: Band-Ende erreicht.

10 = Datei-Ende erreicht (READ AT END - Bedingung)

21 = Falsche Schlüsselfolge bei Index-Datei
(INVALID KEY - Bedingung)

22 = Schlüssel ist bereits vorhanden
(INVALID KEY - Bedingung)

23 = Kein Satz mit diesem Schlüssel vorhanden
(INVALID KEY - Bedingung)

24 = Datei-Grenze überschritten / Datei-Ueberlauf
(INVALID KEY - Bedingung)

30 = Hardware-Lese- oder Schreib-Fehler oder
Blocklängen-Fehler

34 = Datei-Ueberlauf (bei Organisation sequential)

90 = Datei kann nicht eröffnet werden:
Fehlende Zuteilung, falscher logname oder
falscher OPEN.

91 = Datei nicht abgeschlossen

File-Status-Werte COBOL 74 (Fortsetung)

92 = Datei nicht eröffnet

93 = Zugriff und OPEN-Modus im Widerspruch

94 = Zugriff ausserhalb der Datei

95 = Widerspruch zwischen Angaben im ASSIGN, im Programm und im Disc-Directory. Bei Index-Dateien auch Fehler im Index.

96 = Gerät und OPEN im Widerspruch

97 = Ungültiger Kennsatz (nur bei Magnetband)

98 = Satzlänge im Programm und auf der Datei stimmen nicht überein

99 = auf Disc: Sektor-Lock = Der Sektor ist durch einen andern Prozess gesperrt.
auf Magnetband: Leeres Band erreicht.

9A = Nicht genug Memory verfügbar

9B = Warnung: Index-Datei wird von einem Programm eröffnet, das eine sequentielle Datei verlangt. Die weitere Verarbeitung liest die Datei physisch-sequentiell.

9C = Strom-Ausfall. Der letzte Zugriff ist möglicherweise unvollständig
(Nur bei Drucker oder Magnetband)

9D = Nachspann am Spulenende erreicht
(Nur auf Magnetband)

9F = Zugriff ausserhalb des Datenbereichs

FILE-STATUS - WERTE (COBOL 85)

00 = Zugriff erfolgreich abgeschlossen

02 = Erfolgreicher Zugriff mit erlaubtem Duplicate Key

04 = Falsche Record-Länge

05 = File nicht vorhanden

07 = Kein File mit REEL/UNIT

0E = Tape-Mark gefunden (nur auf Magnetband)

0F = Auf Disc: Warnung: Datei wird voll.
Normalfall: Datei zu mindestens 95 % gefüllt. Im Sysgen kann auch eine andere Limite definiert werden.

auf Magnetband: Band-Ende erreicht.

10 = Datei-Ende erreicht (READ AT END - Bedingung)

14 = Falsche Schlüssel-Grösse

21 = Falsche Schlüsselfolge bei Index-Datei
(INVALID KEY - Bedingung)

22 = Schlüssel ist bereits vorhanden
(INVALID KEY - Bedingung)

23 = Kein Satz mit diesem Schlüssel vorhanden
(INVALID KEY - Bedingung)

24 = Datei-Grenze überschritten / Datei-Ueberlauf
(INVALID KEY - Bedingung)

30 = Hardware-Lese- oder Schreib-Fehler oder
Blocklängen-Fehler

34 = Datei-Ueberlauf (bei Organisation sequential)

35 = Datei nicht zugeteilt

37 = Unerlaubter OPEN-Modus

38 = Datei ist für Zugriffe gesperrt

File-Status-Werte COBOL 85 (Fortsetung)

39 = Konflikt zwischen Datei-Attributen im Programm und im Disc-Directory.
oder:
Warnung: Index-Datei wird von einem Programm eröffnet, das eine sequentielle Datei verlangt. Die weitere Verarbeitung liest die Datei physisch-sequentiell.

41 = Datei ist schon eröffnet.

42 = Datei ist nicht eröffnet.

43 = Erfolgloser Print-Read.

44 = Zugriff ausserhalb der Datei.

46 = Kein Folge-Record gefunden (Read-Befehl).

47 = Lesen nicht erlaubt.

48 = Schreiben nicht erlaubt.

49 = Rewite/Delete nicht erlaubt.

97 = Ungültiger Kennsatz (nur bei Magnetband)

99 = auf Disc: Sektor-Lock = Der Sektor ist durch einen andern Prozess gesperrt.
auf Magnetband: Leeres Band erreicht.

9A = Nicht genug Memory verfügbar

9C = Strom-Ausfall. Der letzte Zugriff ist möglicherweise unvollständig
(Nur bei Drucker oder Magnetband)

9D = Nachspann am Spulenende erreicht
(Nur auf Magnetband)

ALPHABETISCHES STICHWORT-VERZEICHNIS

Hinweis: Die Kapitel 1 bis 19 finden Sie im Band 1,
die Kapitel ab 20 im Band 2.
Der Anhang ist in beiden Bänden enthalten.

Stichwort	**Kapitel/Seite**
#-Funktionen	21/WHE/3
#ERROR (in Controlstrings))	21/NOA/2
@ (Substitutions-Parameter)	21/EXE/5
@ für Kombination von Variablen	21/6 21/EQU/3
$ACCESS	26/1
$COBOL, $COBOL9	30/1
$DINT	8/1
$EDIT	29/1
$HELP	10/1
$LOGUTIL	25/1
$MINT	7/1
$PDP	23/6
$SORT	9/1
$STREAM	6/1
$TCM	24/2
$OBJCONV	27/1
<RET> (SCL-Editor-Befehl)	12/ZUS/1
* (Wildcard-Zeichen)	4/17
* (SCL-Editor-Befehl)	12/ZUS/3
* (View-Befehl)	4/VIE/7
- (View-Editor-Befehl)	4/VIE/7
^ (SCL-Editor-Befehl)	12/I/1
1A (SCL-Editor-Befehl)	12/1X/1
1I (SCL-Editor-Befehl)	12/I/1
1R (SCL-Editor-Befehl)	12/1X/2
< und > (SCL-Editor-Befehl)	12/ZUS/3
? (SCL-Editor-Befehl)	12/ZUS/2
? (View-Befehl)	4/VIE/9
Ä und Ü (Substitutions-Parameter)	21/EXE/5
Ä und Ü (SCL-Editor-Befehl)	12/A/6 12/U/3
Abbruch von Verarbeitungen	11/15 4/ABO/1
Abbruch Background-Jobs	20/ABO/1
Abbruch Control-Strings	21/1 21/EXI/1 21/NOA/1

Stichwort	Kapitel/Seite
Abmelden am Bildschirm	3/6 4 BYE/1
ABORT System-Befehl	4/ABO/1
ABORT (Auto-Spooling)	5/ABO/1
ABORT (Background-Jobs)	20/ABO/1
Abschluss des Systems	11/11
ACCESS ($ACCESS)	26/1
ADJUST ($EDIT-Befehl)	29/AD/1
Aendern Auto-Spoolfile	5/ALT/1
Aendern Background-Jobs	20/ALT/1
Aendern des Physname	4/CHA/1
Aendern von Datei-Spezifikationen	4/CHA/6
Aendern von GROUP-Zuteilungen	22/4
AGAIN ($EDIT-Befehl)	29/AG/1
Aktivieren von Bildschirmen	4/ATT/1
ALTER (Auto-Spooling)	5/ALT/1
ALTER (Background-Jobs)	20/ALT/1
ALTER (SCL-Editor-Befehl)	12/A/1
Alternate Disc	4/SET/14
ANSI-Magnetband-Standard	31/10
ANY als Drucker-Nummer	4/ASS/3
Arbeits-Abbrüche	11/15
ASCII-Code für ITX	Anhang/1
ASCII-Code für PC	Anhang/2
ASCII (View-Befehl)	4/VIE/10
ASSIGN System-Befehl	4/ASS/1
ASSIGN ($EDIT-Befehl)	29/AS/1
ASSIGN (Auto-Spooling)	5/ASS/1
ASSIGN (Multi-Section-Dateien)	22/9
Assign aufheben	4/DEA/1
ATTACH System-Befehl	4/ATT/1
ATTACH (Auto-Spooling)	5/ATT/1
Ausdrucken von Dateien	4/DUM/1 4/MOV/3 4/VIE/20
Automatische Bildschirm-Abmeldung	3/7 4/BYE/2
Auto-Spooling	2/8 5/1
Autostart- (Sysboot-)Controlstring	11/5
Background-Verarbeitungen	20/1
BACKUP ($STREAM)	6/3
Backup-Kopien	4/QBA/1 4/QRE/1
Band siehe unter Magnetband bzw.	Streamer-Band
Batch-Prozess	2/4
Batch-Queue-Flag	20/SET/1
Batch-Queue-Verzeichnis	20/DIS/1
Befehlsfolge (Controlstrings)	21/1, 21/7
Befehls-Format der System-Befehle	4/13
Befehlsauswahl ausgeben	4/DIS/38
Befehlsauswahl setzen	4/SET/4

Stichwort	Kapitel/Seite
Befehlssätze für Bildschirme	4/8
BEGIN System-Befehl	4/BEG/1
Bildschirm abmelden	3/7 4/BYE/1 4/RET/1
Bildschirm inaktivieren	4/DET/1
Bildschirm-Aufteilung	3/4
Bildschirm-Bedienung	3/1
Bildschirm-Dialog, allgemein	3/5
Bildschirm-Zuteilung	2/6
Binär-hex-dezimal-Umrechnung	Anhang/3
Blocklänge	2/12
Block Lock (Sektor Lock)	2/19 4/ASS/7
Boot-Tape erstellen ($STREAM)	6/6
Break-Taste	3/1
BUILD ($STREAM)	6/6
BYE System-Befehl	4/BYE/1
CALL (View-Befehl)	4/VIE/11
Cache (Disc-Cache)	23/24
Cache Memory	4/ASS/4
CHANGE System-Befehl	4/CHA/1
CHANGE ($EDIT-Befehl)	29/CH/1
CHECK System-Befehl	4/CHE/1
COBOL-Compiler	30/1
COBOL9 Compiler	30/1
Cobol-Switch	4/DIS/36 4/SET/13
Compiler (COBOL)	30/1
CONCAT ($EDIT-Befehl)	29/CO/1
CONTINUE-Parameter (Assign)	4/ASS/4
Control-Strings	4/11 21/1
Controlstring mit SCL-Editor-Befehlen	12/ZUS/4
Controlstring-Variable	21/4 21/DIS/3 21/EQU/1
COPY System-Befehl	4/COP/1
COPY (SCL-Editor-Befehl)	12/C/1
Data Analysis ($PDP)	23/15
Data Collection ($PDP)	23/12
Datei-Grösse	2/12 2/13
Datei-Inhalt ausgeben	4/DUM/1 4/VIE/1
Datei-Sezifikationen	2/12
Datei-Strukturen	2/13
Dateien bereinigen nach Abbruch	4/CHE/1
Dateien kopieren	4/COP/1 4/MOV/1
Dateien Löschen	4/DEL/1
Daten-Eingabe in Submit-Batch-Jobs	20/3
DEASSIGN System-Befehl	4/DEA/1
DELETE System-Befehl	4/DEL/1
DELETE ($EDIT-Befehl)	29/DE/1
DELETE (Multi-Section-Dateien)	22/12

Stichwort	Kapitel/Seite
DETACH System-Befehl	4/DET/1
DETACH (Auto-Spooling)	5/DET/1
Dezimal-binär-hex-Umrechnung	Anhang/3
DINT ($DINT)	8/1
Directory (Disc-)	4/DIS/15
Directory (Magnetband-)	4/DIS/32
Disc-Bedarf für das Betriebssystem	31/9
Disc-Cache	23/27 4/ASS/4
Disc-Directory, Ausgabe	4/DIS/15
Disc-Directory, interner Aufbau	31/3
Disc-Dump	4/DUM/1
Disc-Einteilung, Disc-Formate	8/4 31/1
Disc-Initialisierung	8/1
Disc-Typen	31/2
Disc-Zuteilung	4/ASS/1
DISPLAY System-Befehl	4/DIS/1
DISPLAY BATCH (System-Befehl)	20/DIS/1
DISPLAY Directory	4/DIS/15
DISPLAY Directory (Multi-Section-File)	22/13
DISPLAY GROUP	22/8
DISPLAY JCL	4/DIS/37
DISPLAY JCL (System-Befehl)	21/DIS/1
DISPLAY MASK	4/DIS/38
DISPLAY MESSAGE	4/DIS/44
DISPLAY PROGRAMS (System-Befehl)	23/23
DISPLAY SPOOL (Auto-Spooling)	5/DIS/1
DISPLAY STATUS	4/DIS/3
DISPLAY STATUS USAGE (System-Befehl)	23/3
DISPLAY SUBSTITUTE	4/DIS/46
DISPLAY SWITCH	4/DIS/36
DISPLAY 'Meldung'	4/DIS/41
DISPLAY 'Meldung' mit W	21/DIS/2
DISPLAY ($EDIT-Befehl)	29/DI/1
DISPLAY ($STREAM)	6/8
DISPLAY (SCL-Editor-Befehl)	12/D/1
DISPLAY (View-Befehl)	4/VIE/12
DCL ($EDIT-Befehl)	29/DC/1
Drive (Disc)	31/1
Drucker-Eingriffe (Auto-Spooling)	5/3
Drucker-Zuteilung (Auto-Spooling)	5/ASS/1
DUMP ($STREAM)	6/10
DUMP System-Befehl	4/DUM/1
EBCDIC-Code	Anhang/4
EDIT (System-Befehl)	12/9
Editor ($EDIT)	29/1
Editor (SCL-Editor)	12/1
Editor-Workfile ($EDIT)	29/3

Stichwort	Kapitel/Seite
Editor-Workfile (SCL-Editor)	12/13
Eingabe von System-Befehlen	4/19
END (Sort-Parameter)	9/14
END System-Befehl	4/END/1
Entladen von Programmen (UNLOAD)	23/22
Entsperren von Dateien	4/CHA/4
EQUATE (System-Befehl)	21/EQU/1
ERASE (SCL-Editor-Befehl)	12/E/1
ERASE ($EDIT-Befehl)	29/ER/1
ERASE ($STREAM)	6/12
Error-Log	25/1
Ersatz-Spuren-Verzeichnis	4/DIS/30
ESC-Taste	3/4
ESCAPE System-Befehl	11/17
EWF	29/3
EX Utilities siehe unter dem Utility-Namen	
EXECUTE System-Befehl	4/EXE/1 21/EXE/1
EXIT (System-Befehl)	21/EXI/1
Fern-Unterstützung	32/1
File siehe unter Datei	
File Sharing	2/10
File-Status-Werte COBOL 74	Anhang/5
File-Status-Werte COBOL 85	Anhang/7
FIND ($EDIT-Befehl)	29/FI/1
FIND (SCL-Editor-Befehl)	12/F/1
FIND (View-Befehl)	4/VIE/14
FIX (System-Befehl)	4/FIX/1
Format (Disc-Format)	8/3 31/1
Fortsetzungszeile in System-Befehlen	4/13
Freier Platz in einer Datei	4/DUM/10
Gelöschte Dateien im Directory	31/8
Generation	4/16
Gerät aktivieren/inaktivieren	4/SET/12
Geräte-Angabe in System-Befehlen	4/15
Geräte-Status setzen	4/SET/12
Geräte-Uebersteuerung	4/ASS/13
Geräte-Unabhängigkeit	4/ASS/13
Global Programm-Segmente	30/16
GO TO in Controlstrings	21/GO/1
GRID ($EDIT-Befehl)	29/GR/1
GRID (SCL-Editor-Befehl)	12/G/1
GROUP (System-Befehl)	22/4
Grund-Unit (Disc)	8/4
Gruppe mit Wildcard-Zeichen	4/17

Stichwort	Kapitel/Seite
Halt von Verarbeitungen	3/3
Hardware-Error-Log	25/1
Helical Scan Tape	2/20
siehe auch unter Magnetband	
HELP System-Befehl	4/HEL/1
HELP ($HELP-Programm)	10/1
HELP (SCL-Editor-Befehl)	12/H/1
Hex-Ausgabe von Daten	4/DUM/1 4/VIE/10
Hex-Binär-dezimal-Umrechnung	Anhang/3
IF System-Befehl	21/IF/1
Inaktivieren von Bildschirmen	4/DET/1
Index-Datei, New Style	2/16
Index-Datei, Old Style	2/14
Index-Dump	4/DUM/6
Index-Neuaufbau	4/MOV/11-19
Inhalts-Ausgabe Disc	4/DUM/1 4/VIE/1
Inhalts-Ausgabe Magnetband	4/DUM/15
Inhalts-Ausgabe Streamer	6/10
Initialisierung Disc	8/1
Initialisierung Magnetband	7/1
Initialisierung Streamer	6/13
INITIALIZE ($STREAM)	6/13
INPUT (Sort-Parameter)	9/9
INSERT ($EDIT-Befehl)	29/IN/1
INSERT (SCL-Editor-Befehl)	12/I/1
Interaktiver Prozess	2/4
ITX-Systemsoftware	1/1
ITX-WINDOWS	3/7
ITXNET System-Befehl NETEXECUTE	4/NET/1
ITXNET System-Befehl NETLOGON	4/NET/3
ITXNET System-Befehl SET NET	4/SET/9
JCL-CODE	4/DIS/37 21/DIS/1 21/SET/1 21/WHE/4
JUSTIFY (SCL-Editor-Befehl)	12/J/1
Keep Datei-Zuteilung	4/ASS/5
Kennsätze auf Magnetband	7/1 31/10
KEY (Sort-Parameter)	9/9
Kommentar in System-Befehlen	4/13
Konversion von Object-Programmen ($OBJCONV)	27/1
Kopieren von Dateien	4/COP/1 4/MOV/1 4/QBA/1 4/QRE/1 6/3 6/15
Kopieren von Autospoolfiles	4/MOV/10
Kopieren ganzer Platten	4/MOV/22
Kopieren von Multi-Section-Dateien	22/14
Kopieren von/auf Streamer-Band	6/3 6/13

Stichwort	Kapitel/Seite
Label (Magnetband)	7/1 31/10
Label in Controlstrings	21/GO/1 21/IF/2 21/WHE/1
Laden (Vorausladen von Programmen)	23/20
LAN-Anschluss	2/21
Link von Control-Strings	21/EXE/1 21/EXE/3
LIO-Status-Werte	Anhang/5
LIST System-Befehl	21/LIS/1
LOAD (System-Befehl)	23/20
Load Leveling	23/24
Local Programm-Segmente	30/16
Logical Units (Disc)	8/4
Logischer Entscheid in Controlstrings	21/IF/1 21/WHE/1
Logname im ASSIGN-Befehl	4/ASS/3
LOGUTIL ($LOGUTIL)	25/1
Löschen Auto-Spoolfile	5/ABO/1
Löschen einzelner Sections	22/12
Löschen Streamer-Band	6/12
Löschen von Dateien	4/DEL/1
Löschen von GROUP-Zuteilungen	22/4
Löschen von Multi-Section-Dateien	22/12
Magnetband-Directory	4/DIS/32
Magnetband-Dump	4/DUM/15
Magnetband-Initialisierung	7/1
Magnetband-Kennsätze	31/10
Magnetband-Kopien	4/QBA/1 4/QRE/1
Magnetband-Label	31/16
Magnetband-Spezifikationen	2/18
Magnetband-Zuteilung	4/ASS/1
Manuelles Spoolfile	2/9 4/ASS/2 4/ASS/7 4/MOV/9
MARK (SCL-Editor-Befehl)	12/M/1
MARK (View-Befehl)	4/VIE/17
Mask (Bildschirm-Befehlsauswahl)	4/DIS/38 4/SET/2
MASK Befehlsauswahl setzen	4/SET/4
Meldung senden	4/DIS/41
Memory-Verwaltung	2/2
Message Queuing	4/DIS/44 4/SET/7
MINT ($MINT)	7/1
Mirror-Image-Kopie auf Disc/Magnetband	4/MOV/25 4/QBA/4
Mirror-Image-Kopie auf Streamer	6/3 6/15
Modell 10000/85, System-Start	11/8
MOUNT System-Befehl	4/MOU/1
MOVE System-Befehl	4/MOV/1
MOVE Autospoolfile	4/MOV/10
MOVE ganze Platten	4/MOV/22
MOVE PK	4/MOV/10
MOVE RB	4/MOV/11
MOVE SI SO FA	4/MOV/13

Stichwort	Kapitel/Seite
MOVE SYSIN	4/MOV/21
MOVE SYSOUT	4/MOV/20
MOVE ($EDIT-Befehl)	29/MO/1
Multi-Section/Multi-Volume-Files	22/1
N (SCL-Editor-Befehl)	12/N/1
N (View-Befehl)	4/VIE/17
Native-COBOL-Compiler	30/1
NCR Fernunterstützung	32/1
NETEXECUTE System-Befehl	4/NET/1
NETLOGON System-Befehl	4/NET/3
Netzwerke (vgl. auch ITXNET)	2/21
Newline-Taste	3/1
NOABORT (Controlstrings)	21/NOA/1
Normal Befehlssatz	4/8
NUMBERS (SCL-Editor-Befehl)	12/N/1
OBJCONV ($OBJCONV)	27/1
Object-Programm-Konversion ($OBJCONV)	27/1
Object-Programm-Struktur	30/15
OPTION (SCL-Editor-Befehl)	12/O/1
OPTION (View-Befehl)	4/VIE/18
OPTION (Sort-Parameter)	9/13
OUTPUT (Sort-Parameter)	9/12
Packnummer	4/17
PAGE (SCL-Editor-Befehl)	12/P/1
PAGE (View-Befehl)	4/VIE/24
Passwort ($ACCESS)	26/1
Patchen auf Disc	4/FIX/1
PC als Terminal	3/1 3/7
PC-ASCII-Code	Anhang/2
PDP ($PDP)	23/6
Performance diagnostic Package $PDP	23/6
Physname	4/16
Place-Parameter (Multi-Section-Dateien)	22/9 4/DIS/29
Platten-Einteilung	31/1
POSITION ($EDIT-Befehl)	29/PO/1
PRINT ($EDIT-Befehl)	29/PR/1
Prioritäten der Prozesse	23/18
Private Dateien	4/ASS/6
Priviledged Befehlssatz	4/8
Programm starten	4/EXE/1
Protokoll-File (Submit-Batch-Jobs)	20/4
Prozess- Prioritäten	23/18
Prozess	2/4
Prozess-Identifikation xxx.yy	4/17

Stichwort	Kapitel/Seite
QBACKUP System-Befehl	4/QBA/1
QRESTORE System-Befehl	4/QRE/1
QUIT ($EDIT-Befehl)	29/QU/1
QUIT (SCL-Editor-Befehl)	12/Q/1
Raster ausgeben (SCL-Editor)	12/G/1
Read only Datei-Zuteilung	4/ASS/6
Reaktivieren Background-Jobs	20/ALT/1
Rebuild von Index-Dateien	4/MOV/10-19
Recordlänge	2/12
Rekonstruieren System-Disc	28/1
Remote Attach NETLOGON	4/NET/3
Remote Execute	4/NET/1
Remote Support	32/1
REMOVE System-Befehl	4/REM/1
Reorganisieren von Index-Dateien	4/MOV/10-19
REPEAT (SCL-Editor-Befehl)	12/R/1
REPEAT (View-Befehl)	4/VIE/24
RESEQUENCE ($EDIT-Befehl)	29/RE/1
RESTORE ($STREAM)	6/15
RESUME System-Befehl	4/RES/1
RESUME (Auto-Spooling)	5/RES/1
RETURN-Taste	3/1
RETENSION ($STREAM)	6/19
RETURN System-Befehl	4/RET/1
SAME-Parameter (Multi-Section-Dateien)	22/10
Satzlänge	2/12
SAVE ($EDIT-Befehl)	29/SA/1
Schlüssel (Index-Datei)	4/DUM/6 4/MOV/10-20
Schlüssel (Sort)	9/1 9/9
SCL-Editor	12/1
SCL-UNIT Geräteangabe	4/15
Scratch-Datei	4/ASS/6
SEARCH ($EDIT-Befehl)	29/SE/1
SEC-Parameter (Multi-Section-Dateien)	22/9
Section einer Datei	4/DIS/27 22/2
Section Label auf Disc	31/5
Security-System ($ACCESS)	26/1
Seitenhöhe von Listen	4/ASS/4 4/SET/10
Sektor Lock	2/19 4/ASS/7
Selektion (SCL-Editor)	12/17
SET System-Befehl	4/SET/1
SET BATCH System-Befehl	20/SET/1
SET JCL System-Befehl	21/SET/1
SET LINES System-Befehl	4/SET/3
SET MASK System-Befehl	4/SET/4
SET MESSAGE System-Befehl	4/SET/7
SET NET System-Befehl	4/SET/9
SET PAGE System-Befehl	4/SET/10

Stichwort	Kapitel/Seite
SET PRIORITY System-Befehl	23/19
SET STATUS System-Befehl	4/SET/12
SET SWITCH System-Befehl	4/SET/13
SET UNIT System-Befehl	4/SET/114
SET USAGE System-Befehl	23/25
Sicherheits-Kopien (Disc)	4/COP/1 ff 4/MOV/1 ff
Sicherheits-Kopien (Magnetband)	4/QBA/1 4/QRE/1
Sicherheits-Kopien (Streamer)	6/3 6/15
SNAPSHOT	23/6 23/7
Sort (Sortieren von Dateien)	9/1
SPACE-Parameter (Multi-Section-Dateien)	22/10
Spawn-Prozess	2/4
Speicher-Belegung	2/1
Speicherformate auf magn. Datenträgern	31/1
Sperren von Dateien	4/CHA/4
Spool-Verzeichnis (Auto-Spooling)	5/DIS/1
Spoolfile (Auto-Spooling)	5/1
Spoolfile (manuelles)	2/9 4/ASS/2 4/ASS/7
SPREAD-Parameter (Multi-Section-Dateien)	22/10
Sprung im Controlstring	21/GO/1
Start-Controlstring (System-Start)	11/5
Start-Controlstring (einzelner Schirm)	4/ATT/1
Start-Controlstring (einzelner User)	26/12
Start des Systems	11/1
Starten eines Programms	4/EXE/1
Status des Systems	4/DIS/3
Status (File-Status) COBOL 74	Anhang/5
Status (File-Status) COBOL 85	Anhang/7
STOP System-Befehl	11/14
STREAM ($STREAM)	6/1
Streamer-Band	2/20 6/1
STRING ($EDIT-Befehl)	29/ST/1
Struktur des Object-Programms	30/15
SUBMIT (System-Befehl)	20/DIS/1
Submit-Batch-Verarbeitungen	20/1
Submit-Job mit variablen Daten	20/SUB/9
SUBSTITUTE (SCL-Editor-Befehl)	12/S/1
SUBSTITUTE DISPLAY System-Befehl	4/DIS/46
Substitution von Disc-Angaben	4/SET/14
Substitutions-Parameter	21/EXE/5
Subsystem laden/entladen	4/BEG/1 4/ END/1
SUSPEND System-Befehl	4/SUS/1
SUSPEND (Auto-Spooling)	5/SUS/1
Suspendieren von Verarbeitungen	3/3 4/RET/1 4/SUS/1
Swap-File	2/2
Switch	4/DIS/36 4/SET/11
Switchable SCSI Device	2/11 4/MOU/4 5/ATT/2
SYS1, SYS2, SYS3 (Geräteangabe)	2/3 4/15
Sysboot-Controlstring	11/5

Stichwort	Kapitel/Seite
Sysgen-Units auf Disc	8/4
SYSIN/SYSOUT (MOVE)	4/MOV/20-21
System-Abschluss	11/11
System-Auslastung	23/1
System-Befehle	4/1
System-Befehls-Format	4/13
System-Disc	2/1
System-Disc, Platzbedarf	31/9
System-Disc rekonstruieren	28/1
System-Meldungen	10/1
System-Optimierung	23/1
System-Start	11/1
System-Status	4/DIS/3
TAB ($EDIT-Befehl)	29/TA/1
Tabulation ($EDIT)	29/IN/4/1 29/TA/1
Tag-Sort	9/1 9/17
Tages-Start-Controlstring	11/5
TAM starten/abschliessen	24/1
Tape siehe unter Magnetband	
Tastatur (Bildschirm)	3/1
TCM ($TCM)	24/2
TEDWFxxxyy (SCL-Editor)	12/13
Temporäre Datei	4/ASS/6
Text-Editor ($EDIT)	29/1
Text-Editor (SCL-Editor)	12/T/1
TIME System-Befehle	4/TIM/1
TRANSFER (SCL-Editor-Befehl)	12/68
Uebergabe von Variablen	21/7 21/EXE/10
Umbenennen von Dateien	4/CHA/1
Umbenennen von Spoolfiles	5/ALT/1
Umrechnung Dezimal-binär-hex	Anhang/3
UNLOAD (System-Befehl)	23/22
UP Datei-Zuteilung	4/ASS/8
UPDATE (SCL-Editor-Befehl)	12/U/1
USAGE (Set Usage)	23/25
User Security-System ($ACCESS)	26/1
Variable Meldungen senden	21/DIS/6
Variablen in Controlstrings	21/4 21/DIS/3 21/EQU/1
Variablen an Controlsstrings übergeben	21/EXE/10
Variable Daten in Submit-Jobs	20/SUB/9
Verarbeitungs-Halt	3/3
VERIFY ($STREAM)	6/20
VIEW System-Befehl	4/VIE/1
VIEW (SCL-Editor-Befehl)	12/V/1
Vorausladen von Programmen	23/20

Stichwort	Kapitel/Seite
WHEN (System-Befehl)	21/WHE/1
Wildcard-Zeichen	4/17
WINDOW (SCL-Editor-Befehl)	12/W/1
WINDOWS (ITX-WIONDOWS)	3/7
Zeilenbreite Drucker	4/SET/3
Zurückkopieren von Sicherheits-Kopien	4/QRE/1
Zusammensetzen von Variablen	21/6
Zuteilung der Geräte	2/6
Zuteilung Drucker	2/7 4/ASS/1
Zuteilung von Dateien	4/ASS/1
Zwischen-Arbeit	4/RET/2

NCR GmbH
Postfach 100090
8900 Augsburg 1
Telefon 0821/4051