RUHR - UNIVERSITÄT BOCHUM

Arbeitsbericht

des

Rechenzentrums

Direktor: Prof. Dr. H. Ehlich

Nr. 7711 ISSN 0341-0358

Unzulänglichkeiten des TR 440-Programmiersystems und ihre Umgehung

2. geänderte Auflage

von

Rainard Buchmann und Hanno Wupper

Bochum im Juli 1977

ISSN 0341-0358

Rechenzentrum der Ruhr-Universität Bochum
Universitätsstraße 150, Gebäude NA
Postfach 102148

4630 Bochum 1

Zusammenfassung

Bochum im Juli 1974

Dem Benutzer höherer Programmiersprachen offenbaren sich bei der Arbeit mit dem TR 440 - Programmiersystem sehr bald eine Reihe von Unzulänglichkeiten und Unbequemlichkeiten:
Einerseits ist die Kommandosprache trotz des eigentlich sehr schönen Konzepts in ihrer speziellen Realisierung einzelner Kommandos unpraktisch und teilweise sogar widersprüchlich; andererseits verbietet das Programmiersystem den in höheren Sprachen geschriebenen Programmen die Ausnutzung vieler vom Betriebssystem her gebotener Möglichkeiten insbesondere zur Ablaufsteuerung und zur Anforderung von Systemleistungen.
Zur Umgehung dieser Schwierigkeiten wurden am RZ UNI BO eine Reihe von Programmen geschaffen, die in der vorliegenden Schrift gesammelt dargestellt werden.

Bochum im Januar 1977

Bemerkung zur zweiten Auflage

Inzwischen, insbesondere mit der MV18, wurde in diesem Sinne manches am Programmiersystem verbessert, nicht zuletzt auch in der Kommandosprache. Aus diesem Grunde erhält die vorliegende Neuauflage einige Beschreibungen weniger als die erste Auflage.

INHALT

ERSTER TEIL

Gliederung des Problemkreises und Einteilung der beschriebenen Programme

Im Teilnehmer-Rechensystem TNS 440 [1] ergibt sich folgendes Schema für die Anforderung von Systemleistungen:

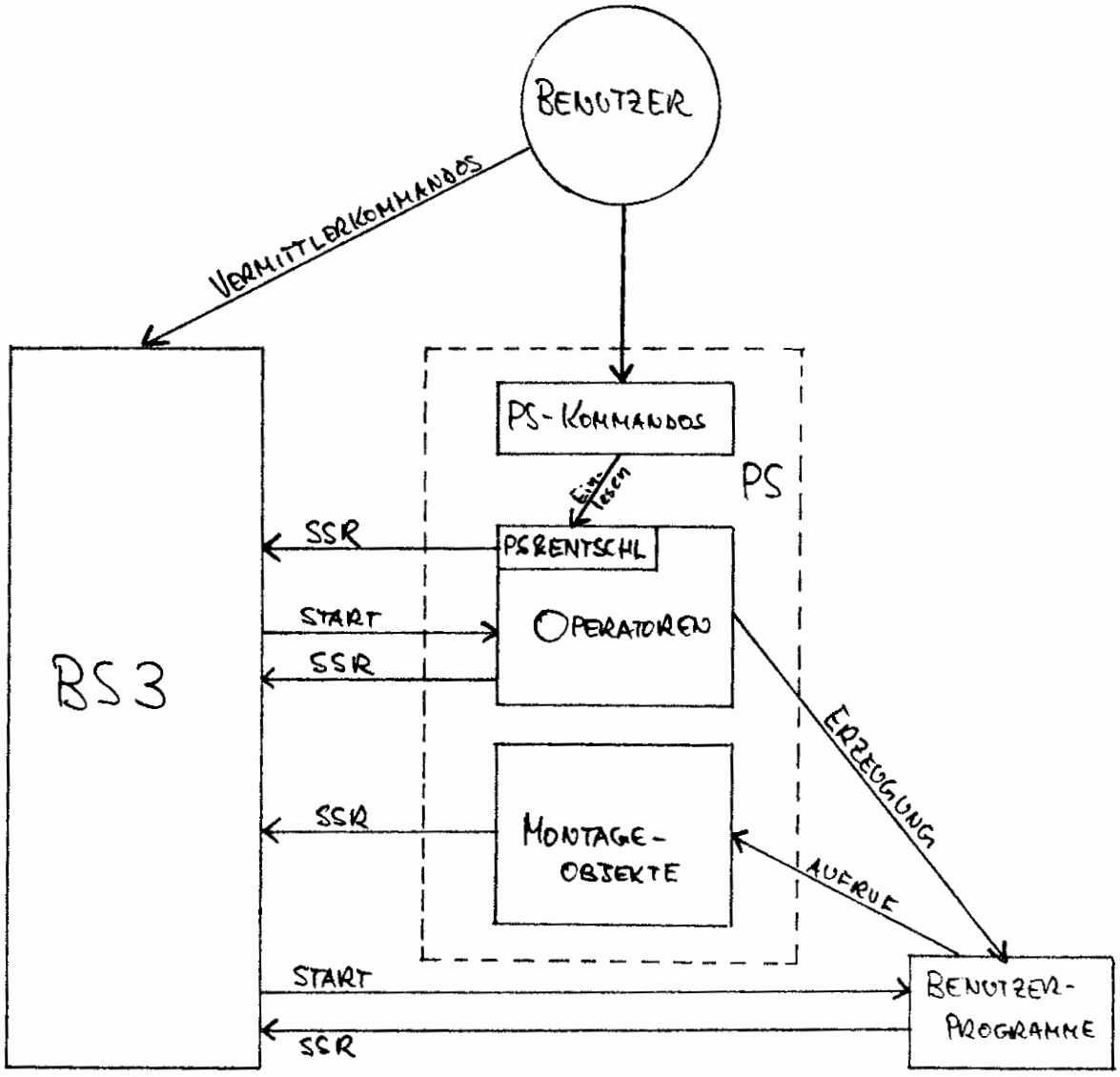

Fig. 1

Das für den Benutzer Wesentliche ist klarer zu erkennen nach einigen Änderungen der Fig. 1:

- Jeder Operatorstart (-Vom für den Benutzer uninteressanten Start des Entschlüßlers durch den Abwickler wird abgesehen-) wird durch einen speziellen SSR-Befehl vom Benutzer oder vom PS verlangt. Die beiden diesem Vorgang jeweils entsprechenden Pfeile werden "kurzgeschlossen"; vom Auftraggeber führt ein Pfeil direkt zum gestarteten Operator.
- Der zwar technisch , aber nicht logisch notwendige PS&ENTSCHL wird entfernt; jedes PS-Kommando bewirkt direkt einen Operatorstart oder einen SSR.
  Der Start des Entschlüßlers wird analog ersetzt durch direktes Geben von Kommandos.
- Der bei der Erzeugung eines Benutzerprogramms durchlaufene Weg über Kommandos, Übersetzer u.s.w. wird durch einen direkten Weg vom Benutzer zu seinen Programmen dargestellt.

Hieraus ergibt sich Fig. 2.

Zu beachten ist, daß im Programmiersystem natürlich nicht alle denkbaren Möglichkeiten zum Start eines Operators oder zum Durchreichen eines SSR vorgesehen sind; über einzelne Wege lassen sich nur Teilleistungen anfordern. Dies ist in Fig. 2 durch eine kleine Wolke an den betr.

Pfeilspitzen angedeutet. Von einem in Maschinensprache geschriebenen Benutzerprogramm aus sind natürlich alle Leistungen des TNS 440 erreichbar.

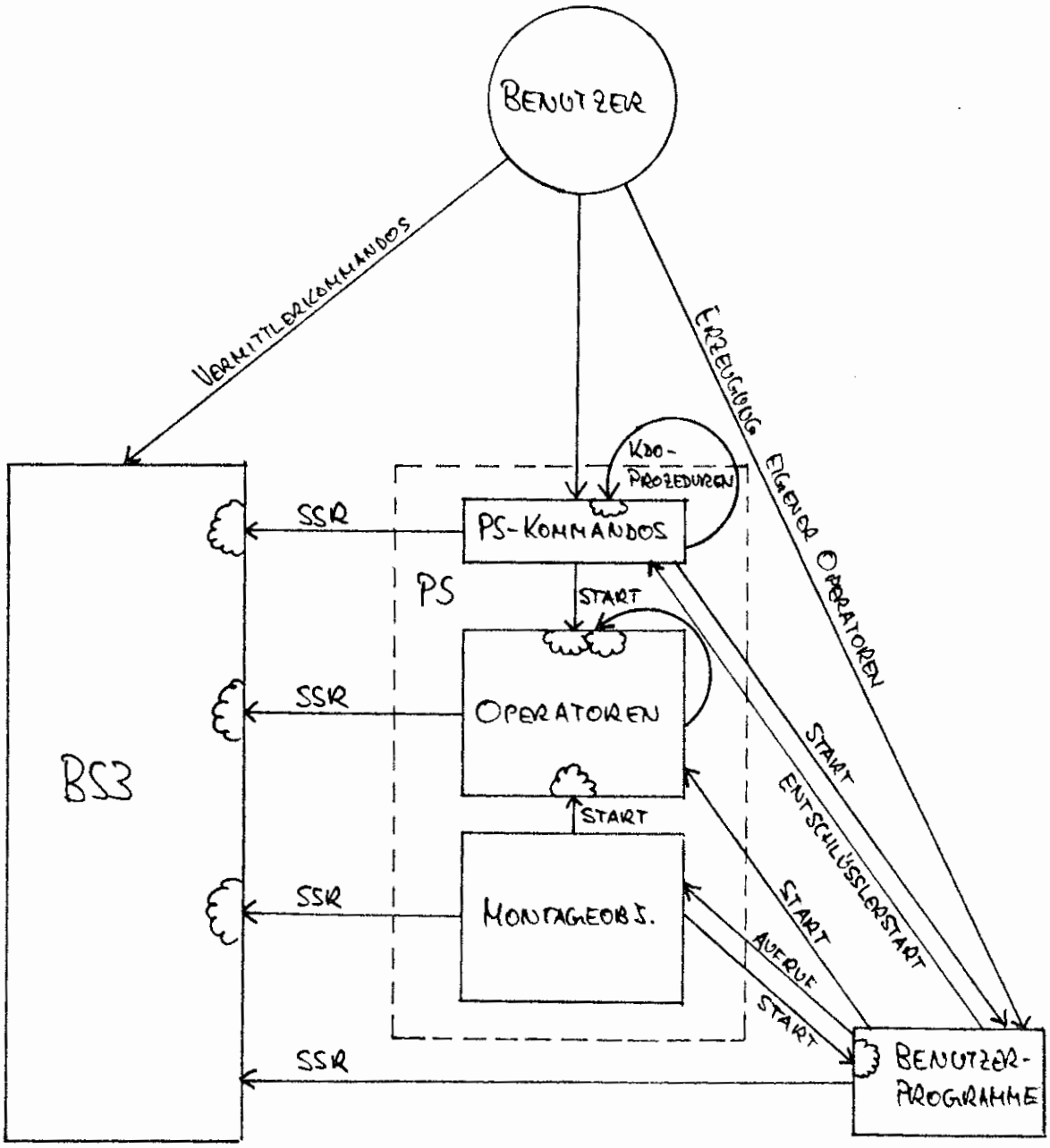

Fig. 2

Das Schema "vereinfacht" sich drastisch, wenn man auf solche Maschinensprache - Programme verzichtet und nur die Möglichkeiten betrachtet, die einen Benutzer höherer Programmiersprachen zur Verfügung stehen (Fig. 3).

Es fällt sofort auf, daß die Möglichkeiten des Benutzerprogramms, oder auch die Möglichkeiten, dieses zu starten und zu steuern, sehr beschränkt sind.
Bei dem vom Hersteller gelieferten PS [2]
ergeben sich eine Reihe von Schwierigkeiten:

1. Anders als für die PS - Operatoren gibt es für Benutzerprogramme nur unzureichende Steuerungs- oder Parametrisierungsmöglichkeiten beim Start, da dieser nur über ein starres Starte-Kommando bzw. eine analoge Prozedur erreichbar ist. Die Steuerung des weiteren Ablaufs durch Benutzerprogramme ist nur über Wahlschalter möglich.

2. Eine Steuerung über wichtige interne Zustandsgrößen des Systems ist kaum möglich, da Benutzerprogramme nur auf eine winzige Teilmenge Zugriff haben.Ähnliches gilt für die Kommandos.
   Überhaupt ist Fig. 3 in keiner Weise kommutativ:
   Der Sortier-Operator ist mehr oder weniger das einzige so wohl über Kommandos als auch über UP-Aufruf startbare Programm des Programmiersystems;
   auf BS 3 - Ebene gilt analoges fast nur für die Wahl-

schalter und einzelne Zustandswahlschalter.

(Natürlich ist immer eine gewisse Steuerung über Dateien möglich, aber doch sehr unhandlich.)

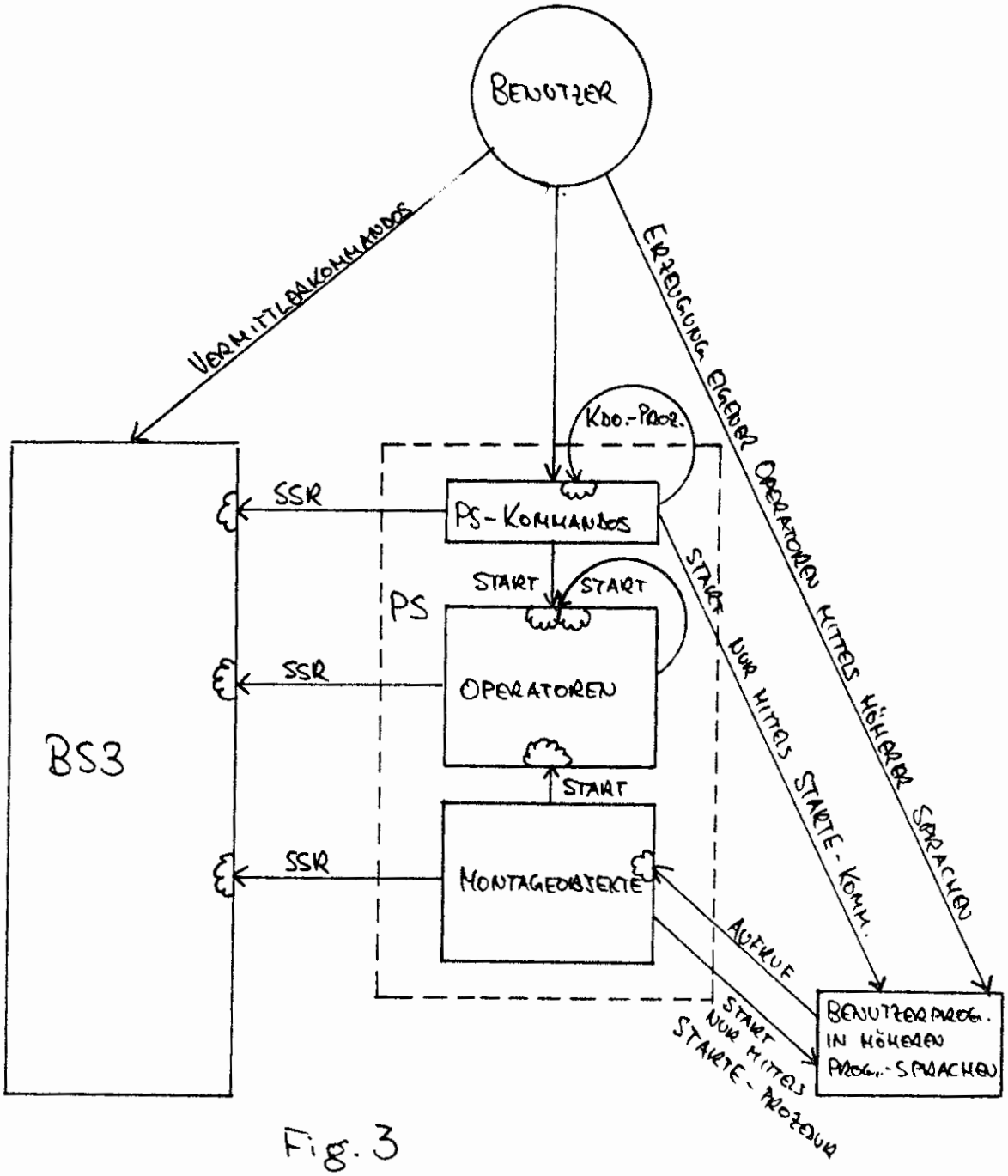

Fig. 3

1. STEUERUNG DES ABLAUFS

1.1. PARAMETRISIERUNG VON BENUTZERPROGRAMMEN BEIM START.

In höheren Sprachen geschriebene Benutzerprogramme können nur über das STARTE-Kommando bzw. das UP STARTE gestartet werden; vom Programm verarbeitbare Information läßt sich dabei nur über die Spezifikation DATEN, also als Fremdstring übergeben, was oft als unpraktisch empfunden wird.

Beispiel:

```
»STARTE,HP1,DATEN=/
TITEL
1.VERSUCHSPERSON                Parameter zur
INTERPOLATION:LINEAR            Programmsteuerung
ANZAHL:␣13
DATEN
  :
  /    } eigentliche Daten
  :
```

1.1.1. Bessere Ausnutzung der Möglichkeiten des STARTE-Kommandos bzw. des UP STARTE.
Die folgenden Programme erlauben Zugriff auf weitere Spezifikationen, die eigentlich anderen Zwecken dienen, aber bei ausgetesteten Programmen nicht unbedingt benötigt werden und sich deshalb zur Informationsübergabe eignen. Zweckmäßig wird ein dem Problem angepaßtes Kommando über eine Kommandoprozedur auf STARTE abgebildet.

1.1.1.1. OLNAME ermöglicht Zugriff auf den Laufnamen (Spezifikation LAUF); Standardname

1.1.1.2. AKTIV (gleichnamige Spezifikation (leider beim UP STARTE nicht angebbar)); Normalstring

1.1.1.3. NUEBWS Überwacher-Seitenschranke (Spezifikation UEBWS); natürliche Zahl

1.1.1.4. NUMMD DNummer-Liste (Spez. DNUMMER); Zahlenpaare der Form nUm

Beispiel:
Das Programm HP1 wird in geeigneter Weise mit etwa folgenden Anweisungen versehen:

```
CALL AKTIV (TITEL)
CALL ØLNAME (INTP)
ANZ=NUEBWS(X)
  :
  :  } Auswertung von TITEL,INTP,ANZ
  :
```

Dann ist nach der Deklaration

```
¤*UNTERSUCHE (TITEL,INTERPOLIERE,ANZAHL,DATEN)
 ¤STARTE,HP1,LAUF=*INTERPOLIERE,AKTIV=*TITEL,
   UEBWS=*ANZAHL
 ¤**
```

folgendes schöne Kommando möglich:

```
¤UNTERSUCHE,TITEL=1.VERSUCHSPERSON,INTERP.=LINEAR,
 ANZ.=13, DATEN=/
 :
 :  } eigentliche Daten
 :
```

1.1.2. Operatorstart durch beliebig definierte Kommandos

Ein völlig beliebiger Startsatz kann durch ein Benutzerprogramm in einer höheren Programmiersprache nicht ausgewertet werden; eine hinreichend variable und elegante Steuermöglichkeit läßt sich erreichen durch Zwischenschaltung eines geeigneten Operators:

RB&BASTEL läßt sich durch ein beliebig definiertes Kommando starten. Der Operator legt seinen Startsatz aufbereitet und formatisiert in einer Texthaltungsdatei ab, die vom nachfolgenden Benutzerprogramm leicht ausgewertet werden kann.

Das Benutzerprogramm kann man dann wahlweise durch den Operator RB&BASTEL starten lassen oder in einer Kommandoprozedur durch das STARTE-Kommando; dabei ergibt sich der Vorteil, daß beim Austesten solcher Programme die Leistungen der Kontrollprozedur verfügbar sind - z.B. die Kontrollereignisverwaltung etc.

Beispiel:
Folgendes Kommando kann durch ein Programm in einer höheren Programmiersprache leicht bearbeitet werden:

```
¤PLOTTE , TABELLE = BUSCH.MAX&MORITZ(2.0) ,
          VON = 3.1415926535 ,
          BIS = 1 . 6 3 E 4  ,
          INTERPOLIERE = Q U A D R A T I S C H  ,
          BREITE = 58 ,
          HOEHE = 80 ,
          XACHSE = /   X  -  A C H S E  ¤/  ,
          YACHSE =     Y  -  A C H S E
```

Dabei bedeutet:

TABELLE=	Name einer Datei, in der Koordinaten einer zu plottenden Meßreihe stehen
VON=	Anfangswert des auszuplottenden Intervall
BIS=	Endwert
INTERPOLIERE=	Interpolationsmethode
BREITE=	Breite der Zeichnung in cm
HOEHE=	Höhe der Zeichnung in cm
XACHSE=	Beschriftung der Ordinate , als Fremdstring angegeben, wenn Leerzeichen mit ausgewertet werden sollen sonst als Normalstring.

YACHSE= Beschriftung der Ordinate (wie bei XACHSE)

Die Definition des Kommandos kann dabei folgendermaßen lauten:

```
¤DEFINIERE, P L O T T E , RB&BASTEL ,
SPEZ. = TABELLE(NL,DT) ' VON(NL,N) ' BIS(NL,N)
      ' INTERPOLIERE(NL,SN,STD) ' BREITE(NL,NZ4) ' HOEHE(NL,NZ4)
      ' XACHSE(NL,N,F) ' YACHSE(NL,N,F)  '  PROGRAMM(NL,SN) ,
EINGANG = 53  ,  OBLIGAT = 4

¤*PROGRAMM(PLOTTE) = BENUTZERHP , *INTERPOLIERE(PLOTTE) = -STD-
```

Die Texthaltungsdatei "BASTEL&DATEI", die dabei vom Benutzerprogramm "BENUTZERHP" ausgewertet werden müßte, sähe bei obigem Aufruf des Kommandos "PLOTTE" so aus:

```
000001   10
000002    1
000003    2 25BUSCH.MAX&MORITZ(0002.00)
000004    1
000005    6 123.1415926535
000006    1
000007    6  61.63E4
000008    1
000009    5 11QUADRATISCH
000010    1
000011    3  5   58
000012    1
000013    3  5   80
000014    1
000015    7 21   X - A C H S E
000016    1
000017    6  7Y-ACHSE
000018    1
000019    5 10BENUTZERHP
000020    1
000021    3  5   53
```

Zeichenzahl

Leerzeichen werden entfernt

Im Fremdstrings bleiben Leerzeichen erhalten

Eingangsnummer

1.2. ABLAUFSTEUERUNG DURCH BENUTZERPROGRAMME

Außer durch die Zustände gewisser Systemgrößen wie Wahlschalter, wird der Ablauf eines Auftrags dadurch beeinflußt, ob ein "Operatorlauf mit Fehler beendet" wird. Bei Benutzerprogrammen ist dies nicht steuerbar; nur die vom Benutzer ungewollten Programmierfehler erzeugen (über Alarme etc.) Fehlerabbruch.

BEENDE mit seinen Eingängen ABBRUC u.s.w. ermöglicht auch Benutzerprogrammen einen Fehlerabbruch z.B. bei Versorgungsfehlern.

2. ERREICHBARKEIT VON SYSTEMLEISTUNGEN

2.1. ERWEITERUNG DER MÖGLICHKEITEN VON BENUTZERPROGRAMMEN

Die hier genannten Unterprogramme ermöglichen Benutzerprogrammen die Anforderungen von sonst nur in Maschinensprache erreichbaren Leistungen.

2.1.1. Durchschaltung von PS-Kommandos

Eine Reihe von auch für Benutzerprogramme interessanten Tätigkeiten lassen sich nur auf Kommandoebene ansprechen:

2.1.1.1. KOMMDO ermöglicht jedem Benutzerprogramm die Ausführung beliebiger PS-Kommandos (durch internen Start des Entschlüßlers).

Beispiel: CALL KOMMANDO ('MELDE,UHR')

2.1.1.2. Für einzelne Tätigkeiten gibt es spezielle Unterprogramme, die gewissen Kommandos mehr oder weniger entsprechen, aber zwei wichtige Vorteile vor der Verwendung von KOMMDO bieten: Eine Operatorlauf-Stufe wird durch Umgehung des Entschlüßlerstarts eingespart und die vom Kommando benötigten Dateien werden vom Benutzerprogramm aus nicht über ihren Namen, sondern über die symbolische Gerätenummer angesprochen.

2.1.1.2.1.	DATEI	entspricht dem gleichnamigen Kommando
2.1.1.2.2.	DRUCKE	entspricht den Kommandos DRUCKE, STANZE und ZEICHNE
2.1.1.2.3.	TUE	entspricht dem gleichnamigen Kommando
2.1.1.2.4.	RESERV	entspricht dem Kommando RESERVIERE
2.1.1.2.5.	LOEDAT	erbringt die Leistung von LOESCHE,DATEI=...
2.1.1.2.6.	BØ&LFD	erbringt die Leistungen der Kommandos EINSCHLEUSE und ABMELDE
2.1.1.2.7.	UP&PROTOKOLL	wirkt wie STARTE,BØ&PROTOKOLL,...

2.1.1.3. Soll ein Benutzerprogramm Dateien ansprechen, die nicht im STARTE-Kommando aufgeführt werden, muß die Datei-Angabe dynamisch erweitert werden:

NDATEI erlaubt es, nachträglich Zuordnungen von Dateien zu symbol. Gerätenummern in den Startsatz einzutragen.

2.1.2. Eröffnung anderer Systemleistungen durch Unterprogramme

2.1.2.1. Andere Systemleistungen sind so beschaffen, daß sie von einem Benutzerprogramm direkt per SSR angefordert werden müssen

2.1.2.1.1. KENDAT liefert die Informationen der SSR 4 0, SSR 4 28, SSR 253 32 und SSR 4 32 aus, das sind fast alle wichtigen Zustandsgrößen des Auftrags.

2.1.2.1.2. SIGNAL erlaubt das Abfragen der (vom Operateur gesetzte) RZ-spezifischen Signale.

2.1.2.1.3. VERDRG stößt eine sofortige Ausgabe auf Konsole ohne anschließende Eingabe-Anforderung an.

2.1.2.1.4. ALSETZ und
ALTEST ermöglichen eigene Abhandlung von Alarmen. (Ohne diese Hilfsprogramme wird der Lauf eines Benutzerprogrammes beim Auftreten eines Alarms unbedingt abgebrochen).

[2.1.2.1.5. Zu nennen wären auch die Pakete BOGOL-TAS und BOTRAN-TAS, die wegen ihres großen Umfanges an diesem Orte nicht beschrieben werden.]

2.1.2.2. Einige PS-Montageobjekte bieten Eingänge zur weiteren Steuerung, die bisher nur mittels Maschinensprache angesprungen werden konnten.

2.1.2.2.1. JAJA erlaubt unter gewissen Umständen Simulation einer unerwünschten Eingabe in Zusammenhang mit PLØSIG.

2.1.3. Einige PS-Montageobjekte werden an jedes Benutzerprogramm zwangsweise anmontiert. Nicht immer ist man mit allen ihren Leistungen zufrieden, weshalb Alternativversionen geschaffen wurden.

2.1.3.1. S&GZK erkennt Eingabeende von Konsole

2.1.3.2. S&OPZEIT unterdrückt die manchmal unerwünschten Anfangs- und Endemeldungen.

2.1.3.3. S&DPRO unterdrückt die Auflistung der geänderten Dateien.

2.1.3.4. F&STOP unterdrückt die Worte STOP und PAUSE, nicht aber die in der FTN-STOP- oder -PAUSE-Anweisung angegebene Textkonstante.

2.1.3.5. SYMBOL erlaubt das Zeichnen eines erheblich vergrößerten Zeichenvorrats (alle belegten ZC1-Zeichen und kyrillische Buchstaben).

2.1.4. BO&PAGING ermöglicht es dem Benutzerprogramm, bis zu 1790 K virtuellen Kernspeicher zu benutzen. Eine Beschreibung ist an diesem Orte nicht möglich.

2.2. ERWEITERUNGEN DER MÖGLICHKEITEN AUF KOMMANDOEBENE

2.2.1. Operatoren, die mittels STARTE gestartet werden, also keiner speziellen Kommandos befürfen.

2.2.1.1. BO&PRØTØKØLL gestattet eine Weiterverarbeitung des Ablaufprotokolls: Es läßt sich z.B. durch diesen Operator in eine Datei ablegen.

2.2.2. Operatoren, die neue Kommandos realisieren

2.2.2.1. BØ&TUE erstellt eine Kommandofolge oder einen vollständigen (KOMSYS-) Auftrag aus einer Datei oder einem Fremdstring.

Die Datei kann auf einem externen Medium liegen und braucht nicht eingeschleust zu sein.

2.2.2.2. ZUSTAND ermöglicht zahlreiche Abfragen und Ab-Prüfungen sowie Verdrängung mit sofortiger Teilausgabe und Abbruch des Auftrages auf Kommandoebene. Etwaige Ergebnisse werden zur Weiterverarbeitung auf Wahlschalter abgebildet.

2.2.2.3. BO&KOMVOR stellt die Möglichkeiten, die das globale Kommunikationssystem KOMSYS bietet, dem Benutzer auf Kommandoebene zur Verfügung.

2.2.2.4. TRANSPORT erlaubt schnelles Kopieren von Dateiinhalten.

2.2.2.5. ERZEUGE erlaubt - für Spezialzwecke - Operateur Anfragen und Verdrängung eines Auftrages.

2.2.2.6. DTSENDE ermöglicht den Austausch beliebiger Dateien zwischen verschiedenen Aufträgen.

2.2.2.7. SENDE ermöglicht das Senden von Texten und Kommandos an fremde Aufträge.

2.2.2.8. EMPFANGE dient zum Empfangen von SENDE/DTSENDE - Sendungen, also zum Auswerten solcher Sendungen, zum reinen Kooridinieren zwischen verschiedenen Aufträgen oder einfach zum Warten ohne bestimmtes Ereignis.

2.2.2.9. DIALOG dient zum koordinierten Dialog zwischen verschiedenen Gesprächen.

2.2.3. Aufwärtskompatible Erweiterungen bereits vorhandener Kommandos

2.2.3.1. ZEICHNE Im Hinblick auf den Betrieb neuartiger graphischer Ausgabegeräte wurde das ZEICHNE-Kommando erweitert.

3. ERWEITERUNG DER KOMMANDOSPRACHE ZUR UMGEHUNG SYNTAKTISCHER SCHWIERIGKEITEN

ERZEUGE ermöglicht auf Kommandoebene

- Verkettung von Teilwerten
- Auswahl eines Teilwertes aus einer Liste durch Indizierung
- Interpretation eines Strings als Formel und Berechnung u.v.a.m.

Die Ergebnisse können zwecks Weiterverarbeitung internen Namen zugewiesen oder einmal als Schleife mit vorgebbaren Abbruchkriterien als Kommando ausgeführt werden.

Damit ist (-ein ganz einfaches Beispiel-) etwa folgende Prozedur möglich:

```
/* IF (BOOL,THEN,ELSE)
/ERZ.,888,/@+(¤/'*BOOL'/)+1@+¤/,MAL=@
/ERZ.,KØMM.,TEILW.=*ELSE'*THEN,KRIT.=*888
/**
```

Der Aufruf könnte etwa wie folgt aussehen:

```
/IF,2+2≥5,THEN=@KOMMANDO1,ELSE=@KOMMANDO2
```

In der Praxis wird die Bedingung dynamisch von einem internen Namen abhängig:

```
//IF,*17'/'GREATER'/'*18,THEN=...
```

Zu beachten ist, daß sich mit den genannten Hilfsmitteln eine Anzahl von Systemleistungen auf mehreren Wegen erreichen lassen und andererseits mehreren verschiedenen Zwecken dienen können:

Da sich Wahlschalter sowohl vom Programm, als auch durch Kommandos abfragen lassen, erlaubt ZUSTAND nicht nur Manipulationen auf Kommandoebene, sondern auch Steuerung von Programmabläufen.

Da KOMMDO eine Verbindung von Programmen zur Kommandosprache herstellt, lassen sich gewisse Tätigkeiten nicht nur über spezielle Unterprogramme vom Benutzerprogramm anfordern.

ZWEITER TEIL

Benutzerbeschreibungen der behandelten Programme in alphabetischer Reihenfolge.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Benutzungsbeschreibung	BØ.E1.03 B
Programm-Thema: Zugriff auf *AKTIV(STARTE) in Fortran-Programmen	Prog.-Name: AKTIV

Gliederung:

1. Zweck des Programms

Bei*AKTIV(STARTE) kann ein beliebiger Normalstring angegeben werden, der bei Programmen ohne Kontrollereignisse vom Programmiersystem nicht ausgewertet wird.

Das Unterprgramm AKTIV liefert die Besetzung dieser Spezifikation als Fortran-String. Dadurch ist Steuerung eines Fortran-Programmes auch über einen Normalstring möglich.

2.1. Programmiersprache: TAS

2.2. Programmierform: Fortran-SUBPØUTINE

3.2. Aufruf:

CALL AKTIV(‹Feld›)

Der String wird auf ‹Feld› abgelegt. ‹Feld› muß lang genug zur Aufnahme des Strings sein; da die Länge eines Startsatzes begrenzt ist, genügt in jedem Falle die Dimensionierung

DIMENSION ‹Feld› (256).

5. Fehlerbehandlung:

Es findet keine Fehlerabprüfung statt.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Benutzungsbeschreibung	BØ.E2.02 B
Programm-Thema: Alarmbehandlung in ALGOL und FORTRAN	Prog.-Name: ALTEST

Gliederung:

1. Zweck des Programms

Dieses Unterprogramm kann immer aufgerufen werden, wenn möglicherweise vorher irgendwann ein Alarm mit Hilfe von ALSETZ abgefangen worden ist. ALTEST fragt dann die gespeicherte Alarmursache ab.

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmform: integer procedure
oder procedure
bzw. INTEGER FUNCTION

3. Handhabung

3.1. Deklaration: a) in ALGOL:

```
integer procedure ALTEST(X); code;
oder    procedure ALTEST(X); code;
```

b) in FORTRAN:

```
ALGOL EXTERNAL ALTEST
INTEGER ALTEST
```

3.2. Aufruf:

Als Funktionsprozedur in arithmetischen Ausdrücken,
z. B.: I := ALTEST(K);
oder als eigentliche Prozedur.

Der Parameter muß eine Variable sein, da in ihr die Alarmursacheals Integer-Zahl abgelegt wird.

Datum: 01. JULI 1974

Dabei bedeuten die möglichen Werte:

0 Ereignisalarm
1 Arithmetischer Alarm
2 Typenkennungsalarm
3 Speicherschutzalarm
4 Überlauf Register U
5 Befehlsalarm
6 Dreierorobenalarm

Als Funktionswert wird übergeben:

-1 wenn noch kein Alarm aufgetreten ist (Die Variable wird dann nicht verändert.).
0 Wenn kein Ereignisalarm aufgetreten ist (d.h. die Variable auf Parameterposition ≠ 0).

Wenn die Alarmursache ein Ereignisalarm ist, die Variable also den Wert 0 erhielt, so spezifiziert der Funktionswert die Art des Ereignisalarmes näher. Dabei bedeuten:

Funktionswert	Alarmart
0	nicht spezifiziert, z.B. bei eintreffendem XAN . bei gesetztem Zustandswahlschalter 4
1	Nettozeitüberschreitung des Abschnitts
2	Überschreitung der Drukkerseitenschranke des Ablaufprotokolls
3	Operateuralarm
4	Mehr als 1024 mal SSR-Fehlerausgänge angesprungen
5	Halt-Befehl von der Konsole eingetroffen (wenn Variante nicht = GS war beim Übersetzen)
6	Entschlüßler soll Gespräch in Grundzustand überführen
7	Sperre für Gemeinschaftsgebiet zu lange gesetzt
8	Nettozeitüberschreitung des Operatorlaufes

Dieselben Werte für die Variable auf Parameterposition und den Funktionswert liefert auch ALSETZ, wenn es mit einer Variablen mit dem Wert 0 als Parameter aufgerufen wird.

Die Ereignisalarme 1 und 2 dürfen nur einmal in einem Abschnitt vorkommen. Dann werden noch Reservezeit und Reserveseiten

für Dumps etc. bewilligt, beim zweiten Auftreten wird dann der Abschnitt sofort abgebrochen.

3.3 Speicherbedarf: siehe ALSETZ

3.4 Zeitbedarf:

4. Arbeitsweise

Es werden nur aus bestimmten Variablen von ALSETZ die Alarmursachen entnommen, und diese Variable wieder auf "noch kein Alarm" gesetzt.

1. Zweck des Programms:

An- und Abmelden von LF-, WSP- und MB-Dateien

2.1. Programmiersprache: TAS

2.2. Programmierform: Montageobjekt für ALGOL60 und FORTRAN

3. Handhabung:

3.1. Deklaration: a) FTN: keine

b) ALGOL: 'PROCEDURE'....(x); 'CODE';
bzw. 'INTEGER''PROCEDURE'....(x); 'CODE'

3.2. Aufruf:

Voraussetzung: Vorhergehender Anfangsaufruf von S&CC [wird von allen Hauptprogrammen gemacht, die mit SPRACHE(UEBERS.)=FTN,ALG60 oder TASR übersetzt wurden] und geeignete Besetzung [vgl. Fehlermeldungen] der Parameter DATEI und eventuell DNUMMER(STARTE).

Parametertypen:

3.2.1. FTN: arithmetischer Ausdruck vom Typ INTEGER*4

ALGOL: arithmetischer Ausdruck, der keine Prozedur aufruft, die nicht nach Algol-Konventionen geschrieben ist, (d. h. die Indexzellen 0 bis 7 werden verändert; z.B. BOGOL-Prozeduren und die BO&LFD-Prozeduren).

3.2.2. a) FTN: Literalkonstante oder Feld, das eine Literalkonstante enthält

ALGOL: -

b) FTN: String im Sinne des FTN-Stringhandling

ALGOL: String oder Feld, das einen String enthält

3.2.3. FTN: label(Anweisungsnummer)

ALGOL: integer label/label/switch-Komponente

Datum: 19. Juni 1974

3.2.4. FTN: INTEGER*4-Variable/-Feldelement
ALGOL: integer-Variable/-Feldelement oder
real-Variable/-Feldelement

Programmbeschreibungen: Beispiele für den Aufruf werden nur in FTN angegeben.

I) TRW: Trägerwechsel. Voreinstellen Träger für folgende LFANL/LFANS-Aufrufe. Vor dem ersten Aufruf von TRW ist <tr> =5 voreingestellt.

Aufruf: CALL TRW(tr[,exdkz])

Parameter 1 [tr] : Er muß vom Typ 1 sein; sein Wert muß zwischen 0 und 8 liegen.
Bedeutung:

(0)	Kernspeicherdatei
(1)	Gebietsdatei auf Platte
(2)	Gebietsdatei auf Trommel
3	MB-Datei, MB(exdkz)
4	WSP-Datei, A-Turm!, W14(<exdkz>)
5	LFD
(6)	
(7)	nicht belegt
8	WSP-Datei, V-Turm, W14(<exdkz>)

(Vgl. auch SSR 253 3/8)

Parameter 2 [exdkz]: Wird für <tr> ≠ 5 ausgewertet, muß vom Typ 2b sein.

II) LFANL: LF-/WSP-/MB-Datei anmelden zum Lesen

Aufruf: a) CALL LFANL (nr[,bkz [,fehl [, label [,dfnr]]]])

b) M=LFANL(nr[,bkz[, fehl]])

Bemerkung: Die Fortran-Syntax erlaubt keine Angabe eines Fehlerlabels im Funktionsaufruf;

in ALGOL ist
M:=LFANL(nr[, bkz[, fehl[, label[, dfnr]]]);
möglich.

Parameter 1 [nr; obligat]: Er muß vom Typ 1 sein.
Sein Wert muß zwischen 1 und 99 einschließlich liegen. Die entsprechende Datei[/STARTE, DATEI=.....] wird in der explizit oder implizit [&STDDB] angegebenen Datenbasis und mit dem eventuell angegebenen Paßwort zum Lesen angemeldet.
DNUMMER wird dabei ausgewertet.

Parameter 2 [bkz; optional]: Ist er vom Typ 1 und gleich 0, so wird die anzumeldende Datei im Standardkatalog (vgl. Kommando INFORMIERE,DATEI=) gesucht.

Beim Typ 2a werden bis zu 6 Zeichen als BKZ übernommen; das Auftreten eines Leerzeichens oder Ignores beendet das BKZ; daher darf das Fortran-Literal nicht mit einem Leerzeichen oder Ignore beginnen. Handelt es sich um eine Literalkonstante, so werden außerdem höchstens soviel Zeichen ausgewertet, wie die Literalkonstante lang ist.

Typ 2b ist auch zulässig: Ein Fortran-String wird dadurch identifiziert, daß sein erstes Element Typenkennung 2 oder 3 hat; dieses wird unverändert als BKZ genommen insbesondere werden Leer-Zeichen nicht ausgeblendet. Von einem Algol-String werden bis zu 6 Zeichen ausgewertet.

Parameter 3 [fehl; optional] : Er muß vom Typ 4 sein. Nach einwandfreiem Ablauf des Anmeldens wird er mit 0 besetzt, ansonsten mit der geeigneten Fehlernummer [siehe dort.] Wenn LFANL als FUNCTION aufgerufen wird, so ist der Wert der FUNCTION derselbe wie der von fehl.

Parameter 4 [label; optional]: Er muß vom Typ 3 sein. Falls beim Anmelden ein Fehler auftritt, so wird das Programm bei der angegebenen Anweisung fortgesetzt; fehlt der Parameter 4, so wird das Programm in jedem Fall mit der nächsten Anweisung fortgesetzt.

Parameter 5 [dfnr; optional]: Er muß vom Typ 1 sein und entspricht der (Datei-)Folgenummer p in EINSCHLEUSE,TRAEGER=MB(exdkz)1.p; die Abschnittsnummer ist stets gleich 1.
Für Zahlen 0 wird 0 eingesetzt.

Achtung: Ist bei einem Aufruf ein optionaler Parameter besetzt, so müssen alle vorhergehenden Parameter besetzt sein!

III) LFANS: Datei anmelden zum Schreiben

Aufruf: a) CALL LFANS(nr[, bkz[, fehl[, label[, dfnr]]])

b) M=LFANS(nr[, bkz[, fehl]])

Die Bedeutung der Parameter ist wie in I), nur wird die Datei zum Schreiben angemeldet.

IV) LFAB: LF-Datei[-en] abmelden

Aufruf: a) CALL LFAB (nr[, fehl[,label]])

b) M=LFAB (nr[, fehl])

Die Bedeutung der Parameter ist wie in I), nur wird die LF-Datei abgemeldet; daher ist bkz und dfnr unnötig. Außerdem kann Parameter 1 den Wert 100 haben. In diesem Fall werden alle eingeschleusten Dateien aus der &STDDB und allen anderen Datenbasen abgemeldet, die nicht entsprechend kreiert worden sind (vgl. SSR 253 1, DBA=1).

Achtung: Bevor man eine Datei abmeldet, die durch die

Fortran- oder Algol-E/A bearbeitet wurde, muß man sie von der Bearbeitung abmelden. Dies geschieht bei Bearbeitung durch die Fortran-E/A mit CLODA, bei der Algol-E/A mit CLOSE. Das gleiche gilt, wenn man eine zum Lesen angemeldete Datei zum Schreiben anmeldet, oder umgekehrt.

Beispiel:

```
/1XBA,BEN=471111KFD.
/UE.,,FTN,Q.=/
        :
      CALL  LFANS (12,'RZ',I,&999)
        :
      CALL  LFANL(14,O,J)
        :
      CALL  CLODA(12)

      CALL  CLODA(14)

      CALL  LFAB(100)

      STOP

 999  WRITE(6,4711)      I

4711  FORMAT('KEINE ANMELDUNG:FEHLER',I8)

      END

/MONT.,DATENBASIS,DAB,STARTE,DNUMMER=12U13,
  DATEI=14-DAB.TEST'13-ABC(2.3)-PASS
```

Es werden RZ.ABC(2.3)-PASS in der &STDDB zum Schreiben und KFD.TEST in DAB zum Lesen angemeldet. Im Fehlerfall ist nach dem entsprechenden Aufruf I bzw. J≠0 und im ersten Fall wird die Fehlernummer ausgedruckt. Sonst werden später beide LF-Dateien [und eventuell noch andere] abgemeldet.

5. Fehlermeldungen [vgl. II), Parameter 3]:

0 : kein Fehler

-1 : Dateinummer kleiner als 1 oder größer als 99 bzw. 100

-2 : Dateinummer vergeben durch STARTE, DNUMMER=...

-3 : ␣STARTE,DATEI=-

-4 : ␣STARTE,DATEI=... erhält keine Zuweisung an die unter Berücksichtigung von DNUMMER erhaltene Dateinummer.

-5 : [nur LFANL/LFANS]: bkz beginnt mit Leerzeichen oder Igbore.

Bei allen anderen Fehlermeldungen handelt es sich um SSR-Fehler [vgl. Unterlagensammlung TR 440, Systemdienste BS3]; mögliche Bedeutungen sind z. B.:

LF-Datei nicht vorhanden/angemeldet von anderem Benutzer/falsches Passwort/...

1≤<fehl>≤2048: Standard-SSR-Fehler; <fehl> Fehlerschlüssel

2049 ≤ <fehl> ≤4096: SSR-Fehler beim SSR 253; <fehl> -2048 ist der Fehlerschlüssel

4097 ≤ <fehl>: SSR-Fehler '120', <fehl> -4096 ist der im rechten Halbwort von RQ übergebene Fehlerschlüssel

1. Zweck des Programms

ALSETZ meldet in einem Algol-Programm eine neue Alarmadresse an, so daß der Algol-Programmierer beim Auftreten eines Alarmes selbst entscheiden kann, wie darauf regiert werden soll.

Beispielsweise kann dann mit der Prozedur ALTEST die Alarmursache abgefragt werden.

2. Aufbau

2.1 Programmiersprache: TAS

2.2 Programmform: ALG6o: procedure oder integer procedure
FTN: SUBROUTINE

3. Handhabung

3.1 Deklaration: je nach gewünschter Leistung
als procedure ALSETZ(X); code;
integer procedure ALSETZ; code;
oder integer procedure ALSETZ(X); code;
(In FTN Deklaration unnötig)

3.2 Aufruf: a) als Funktionsprozedur ohne Parameter:
I := ALSETZ;
Diese Aufrufart bewirkt, daß kein Ereignisalarm zugestellt wird, sondern diese

gesammelt werden, und daß bei einem anderen Alarm der Operator an der Unterbrechungsstelle fortgesetzt wird (Achtung: kann u.U. zu einer unendlichen Schleife führen bei einem Alarm!);

b) als eigentliche Prozedur oder Funktionsprozedur mit einem Parameter, der

1.) ein integer label sein kann:
Dann wird nach einem Alarm der Operator bei diesem integer-label fortgesetzt.

2.) eine Variable mit dem Wert = 0 sein kann:
Dann wird nach einem Alarm der Operatorlauf hinter dem Aufruf von ALSETZ fortgesetzt. Diese Aufrufart ist nur sinnvoll als Funktionsprozedur, da ALSETZ dann gleichzeitig als ALTEST fungiert mit entsprechender Wertübergabe (siehe "ALTEST").

c) in FTN :

```
CALL ALSETZ (&1oo)
```

3.3 Speicherbedarf: ALSETZ, ALTEST und SPERRE zusammen:

158 Befehle

24 GW Arbeitsspeicher

14 GW Konstanten

3.4 Zeitbedarf:

4. Arbeitsweise

Die Alarmadresse wird mit dem SSR 020 auf eine Adresse im Unterprogramm ALSETZ gesetzt. Dort wird nach einem Alarm die Alarmursache abgefragt, der Alarm gelöscht (SSR 4 8) und die ursprüngliche Alarmadresse des Operators wiederum als Alarmadresse angemeldet. Die Alarmursache wird außerdem für ALTEST gespeichert, das aber nicht unbedingt aufgerufen werden muß. Dann wird der Operator bei dem label fortgesetzt, dessen Adresse und Hierarchie vorher gespeichert worden waren.

1. Zweck des Programms

Das Unterprogramm hat mehrere Eingänge, die den Operatorlauf sofort beenden bei Aufruf:

a) BEENDE ohne Fehler und ohne Endemeldung

b) ABBRUC mit Fehler und ohne Endemeldung
bzw.
ABBRUCH

c) OPSTOP ohne Fehler und mit Endemeldung

d) OPABBR mit Fehler und mit Endemeldung

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmierform: a) in ALGOL: procedure
b) in FORTRAN: SUBROUTINE

3. Handhabung

3.1. Deklaration:

a) in ALGOL:
```
procedure BEENDE; code;
procedure ABBRUCH; code;
procedure OPSTOP(X); code;
procedure OPABBR(X); code;
```

b) in FORTRAN: nicht nötig

3.2. Aufruf:

a) in ALGOL:
```
BEENDE;
ABBRUCH;
OPSTOP(OLN);
OPABBR(OLN);
```

b) in FORTRAN:
```
CALL BEENDE
CALL ABBRUC
CALL OPSTOP(OLN)
CALL OPABBR(OLN)
```

3.3. Speicherbedarf: 5 Ganzworte Arbeitsspeicher
18 Befehle

3.4. Zeitbedarf: ca. 0.5 msec.

4. Arbeitsweise:

4.1. Zunächst wird die Endebehandlung mittels S&CC angestoßen, danach mit SSR O 12 (bei BEENDE und OPSTOP) bzw. mit SSR O 16 (bei ABBRUCH und OPABBR) der Operatorlauf beendet.

Bei OPSTOP und OPABBR wird vorher noch eine Endemeldung mit S&OPZEIT ausgegeben, wobei der Parameter 'OLN' als Text mit ausgegeben wird statt des sonstigen Operatorlaufnamens. OLN kann in ALGOL ein String oder ein Array sein, das einen String enthält, und in FORTRAN ein String im Sinne des Stringhandlings. Maximal die ersten 12 Zeichen werden ausgewertet.

1. Zweck des Programms

KENDAT liefert Kenndaten des aktuellen Abschnitts und Operatorlaufes in für Algol- und Fortran-Programme aufbereiteter Form aus. Die Information wird in einen Common-Bereich mit dem Namen KENDAT abgelegt.

2. Aufbau

2.1 Programmiersprache TAS

2.2 Programmform:

procedure	in ALGOL
bzw. SUBROUTINE	in FTN
bzw. rekursive Funktion oder Routine	in BCPL

3. Handhabung

3.1 Deklaration:

a) in Fortran nicht nötig
b) in Algol: procedure KENDAT; code;
c) in BCPL: EXTERNAL KENDAT: B. KENDAT, B. KENNDATEN

3.2 Aufruf:

a) in Fortran: CALL KENDAT
b) in Algol: KENDAT;
c) in BCPL: B.KENDAT()

3.3 Speicherbedarf:

292 Befehle
7 Ganzworte Konstanten
154 Ganzworte Arbeitsspeicher

Der Common-Block, in den die Information abgelegt wird, muß folgenden Aufbau haben:

a) in Fortran:

```
COMMON/KENDAT/NKSB, NBGB, NTSB, NPSB,
NDRS, NSBG, NRZS, NANR, NTYP, NGNR,
NSNR, NKSPMX, NTSPMX, NPSPMX,
```

Datum: - 5. JUNI 1974

Dabei ist noch folgendes zu beachten:

Die Angaben unter BEN= und FKZ= werden immer mit Leerzeichen aufgefüllt, d. h. NBENZC und NFKZZC sind mit der Zahl "175" aufgefüllt, alle anderen Felder unter β) sind mit "0" aufgefüllt.

4. Arbeitsweise

4.1 Es werden die Kenndaten des SSR 4 0 benutzt, die BKZ's werden mit dem SSR 253 32 (IS=4) erfragt, das Datum mit dem SSR 4 32 (T=2), die restlichen zur Verfügung stehenden Speicherberechtigungen mit dem SSR 4 28. Dabei ist noch zu berücksichtigen, daß der Wert von NRKSP in Algol meistens kleiner ist als der Wert, den die Prozedur MEMORY liefert, da der aktuelle Wert des Freispeicherpegels nicht berücksichtigt wird, d. h. NRKSP hat als Wert den tatsächlichen freien Kernspeicher, der z. B. durch ein neues Gebiet belegt werden kann, während MEMORY den Wert liefert, der durch ein neues Algol-array im Freispeicher belegt werden kann.

Bemerkung: Für BCPL-Aufruf gilt für die Ablage der einzelnen Größen dasselbe wie beim Fortran-Aufruf. Das heißt also, die Zahlen werden als Festkommagrößen abgelegt, die Strings als Fortran-strings.
Wird KENDAT als Funktionsprozedur aufgerufen, so ist der Funktionswert die Anfangsadresse des Common-Bereiches (besonders für BCPL-Aufrufe gedacht). Diese Anfangsadresse steht für BCPL-Programme aber ebenso in der Zelle mit dem Namen B.KENNDATEN (Halbwort, während alle anderen Größen auch für BCPL in Ganzworten liegen).

Speziell für BCPL wurde noch eine Anzahl von Kontaktnamen geschaffen. Es gibt folgende Kontaktnamen, deren Bedeutung aus den vorigen Seiten ersichtlich ist:

NKSB, NBGB, NTSB, NPSB, NDRS, NSBG, NRZS, NANR, NTYP,
NGNR, NSNR, NKSPMX, NTSPMS, NPSPMX, NOLNZC, NFKZZC,
NBENZC, OLN, FKZ, BEN, NBKZ1, NBKZ2, NBKZ3, NBKZ4,
BKZ1, BKZ2, BKZ3, BKZ4, NRKSP, NRTSP, NRPSP, BMV, BKENN,
MVOP, BDATUM, NTANR, NZEIBR

Größen, die Zahlen enthalten, können dabei z. B. so angesprochen werden: NKSB!1 , da die Kontaktnamen die (Ganzwort-) Adresse der zugehörigen Größen enthalten.

Strings sind normal unter dem Kontaktnamen anzugeben.

Benutzungsbeschreibung BØ E3 .50 B

Programm-Thema: KOMSYS-Leistungen auf Kommandoebene

Prog.-Name: BO&KOMVOR

Gliederung:

Der Operator BO&KOMVOR stellt die Leistungen des TR440-Kommunikationssystems KOMSYS auf Kommandoebene zur Verfügung. Dazu gehört:

Kommando	Leistung
DTSENDE	Austausch von Dateien zwischen Aufträgen
SENDE und EMPFANGE	Koordinierung zwischen Aufträgen und Austausch von Texten, Kommandos etc.
DIALOG	Koordinierter Dialog zwischen Gesprächen
(STARTE)	Allgemeine Warteschlangen-Manipulationen

Datum: .....

Grundsätzlich ist zur Behandlung der Sendungen zu sagen:

Jede Sendung wird über ein frei wählbares KENNZEICHEN identifiziert, das aus einem beliebigen Normalstring besteht, von dem maximal die ersten 12 Zeichen relevant sind.
Dieses KENNZEICHEN wird intern als ABS-NAME der KOMSYS-SSR's benutzt.
Das ABS-FKZ der Sendungen dient zur Identifizierung des Sendungstyps (Datei, Text, Kommandos etc.) und trägt in einigen Fällen noch Zusatzinformation (z.B. den FLULI - Verweis bei Kommandofolgen).

Es gibt zwei verschiedene Formen von Sendungen:

a) Kernspeichersendungen und reine Sendungsköpfe.
   Diese werden immer zu Lasten der Warteschlange gesendet. Da sie beim Übernehmen automatisch gelöscht werden, wird der absendende Auftrag nach der Absendung sofort fortgesetzt - er braucht weder auf eine Rücksendung zu warten noch die Sendung wieder zu löschen.

b) Gebietssendungen werden immer zu Lasten des Absenders gesendet.
   Aus diesem Grund wartet der absendende BO&KOMVOR auf die Rücksendung des Gebietes und löscht die Rücksendung, damit dem Absender die Speicherberechtigung zurückgebucht wird.
   Das bedeutet, daß der absendende Auftrag erst fortgesetzt wird, wenn entweder ein Auftrag die Sendung übernommen und zurückgesandt hat, oder wenn die Verfallszeit der Sendung, die gleichzeitig als Wartezeit für die Rücksendung eingesetzt wird, überschritten wurde.
   In beiden Fällen löscht der absendende BO&KOMVOR die Gebietssendung und damit erhält der Auftrag die abgebuchte Speicherberechtigung zurück.
   Wird also beim Kommando EMPFANGE eine Gebietssendung vorgefunden, so wird zuerst das empfangene Gebiet kopiert, sofort zurückgesandt und dann erst die Information ausgewertet.

An zusätzlichen Objekten wird lediglich der Operator BO&TUE (BO.E5.01) benötigt und zwar beim Kommando SENDE, wenn nicht QUELLE=-STD- angegeben ist, also nicht nur ein Sendungskopf abgeschickt werden soll.
Soll nämlich mittels SENDE eine Textfolge oder eine Kommandofolge gesendet werden, so wird der Operator BO&TUE intern speziell gestartet, wobei die ersten 5 Spezifikationen des Kommandos an diesen weitergereicht werden. BO&TUE erzeugt das zu versendende Gebiet unter Berücksichtigung der Spezifikationen sowie evtl. eine FLULI und meldet beides an den Operator BO&KOMVOR mittels Steuerinformation (SSR 1 Ø) zurück.

Der Operator BO&KOMVOR hat 2 getrennte Aufgabenbereiche

1. Dienstleistungen für Benutzer, die es unter Ausnutzung der KOMSYS-SSR's ermöglichen, Textfolgen, Kommandofolgen oder komplette Dateien zwischen Aufträgen auszutauschen sowie einen Dialog zwischen Gesprächen herzustellen.

2. Er bietet Möglichkeiten allgemeiner Warteschlangenmanipulationen wie KREIEREN, LÖSCHEN, INFORMIEREN etc. sowie gewisse Systemleistungen wie Durchreichen von KFK-Kommandos und Botschaften Absenden, die nur dem Rechenzentrum zugänglich sind.

Diese beiden Bereiche sind logisch dadurch getrennt, daß die RZ-Spezialdienste mit Hilfe des STARTE-Kommandos und die dem Benutzer zugänglichen Dienste mit Hilfe definierter Kommandos angesprochen werden.
Aus Sicherheitsgründen werden die RZ-Dienste hier nicht detailliert beschrieben - eine Beschreibung kann beim RZ der Ruhr-Universität Bochum angefordert werden.

Die Kommandos für den Benutzer werden definiert beim Start des Operators durch

/# STARTE,BO&KOMVOR,DEFINIERE

Im einzelnen gibt es folgende Kommandos, deren präzise Beschreibung in Form von Kommandoblättern ebenfalls in Bochum angefordert werden kann:

1. DTSENDE

Damit ist es möglich, komplette Dateien vom Typ SEQ, RAN, RAM, RAS und PHYS zu versenden, die beim Empfänger in der originalen Form kreiert werden. Einzige Einschränkungen für die Dateien:

a) Bei RAS-Dateien muß der Satzschlüssel im Satz liegen.

b) Sätze länger als 1022 Ganzworte können nicht bearbeitet werden.

c) Eingeschleuste MB-Dateien müssen vom Typ SEQ sein (im Unterschied zu den anderen BO&KOMVOR-Kommandos, bei denen auch eingeschleuste gesicherte RAM-Dateien erlaubt sind).

2. SENDE

Damit ist es möglich, Fremdstrings oder Datei-Bereiche bzw. ganze Dateien als Kommandcfolge (die beim Empfänger ausgeführt wird), als Textfolge (die beim Empfänger ohne Zeilennummern ausgedruckt wird) oder zu Koordinationszwecken nur einen Sendungskopf ohne Informations-Inhalt abzusenden.

3. EMPFANGE

Dieses Kommando dient dazu, alle möglichen von SENDE oder DTSENDE erzeugten Sendungen entgegenzunehmen und ihrem Typ entsprechend zu behandeln.
Dabei ist es möglich, wahlweise auf das Eintreffen einer solchen Sendung zu warten (zu Koordinationszwecken) oder eine Sendung nur zu übernehmen, wenn sie bereits vorhanden ist.

4. AUSDRÜCKE

Mit Hilfe dieses Kommandos kann man Texte aus Fremdstrings oder Dateibereichen ohne Zeilennummern ausdrucken - interessant ist dieser Dienst z.B. bei Benutzung von Kommandoprozeduren zum Ausgeben von beliebigen Texten.
(AUSDRUCKE benutzt keine KOMSYS-SSR's).

5. DIALOG

Dieses Kommando ermöglicht einen koordinierten Dialog zwischen zwei Gesprächen.
Dadurch, daß jeder Dialogteilnehmer sich und seinen Dialogpartner durch eine beliebige, maximal 12 Zeichen lange Zeichenfolge identifizieren kann, ist es möglich, mehrere getrennte Dialoge parallel gleichzeitig zu führen.
Darüberhinaus ist es jederzeit möglich, einen Dialog auch im Zustand "WARTEN AUF ANTWORT" zu beenden.
Durch Anhalten des Abwicklers mittels "/# XAN/# .
und anschließende Eingabe von "HALT/#." auf die Anfrage
"/# /# ARW/# :"
Dies wird dadurch erreicht, daß immer in Einheiten von 30 Sekunden auf das Eintreffen einer Sendung gewartet wird, so daß beim Fortsetzen nach 30 Sekunden der Abwickler ein XAN entgegennehmen kann.

Eine ausführliche Beschreibung dieser Kommandos kann in Form von Kommandoblättern beim RZ der Ruhr-Universität Bochum angefordert werden.

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung BØ.E3.52 B

Programm-Thema: Demand Paging

Prog.-Name: BO&PAGING

Gliederung:

BO&PAGING realisiert ein allgemeines Demand Paging für beliebige Programme. Dabei wird ein bis zu 1920 K Worte großer virtueller Kernspeicher zur Verfügung gestellt.
Kein Programm, das den virtuellen Kernspeicher benutzen soll, muß geändert werden.
Auch Overlay-Programme können das Demand Paging nutzen. Eine ausführliche Beschreibung ist als Rechenzentrumsbericht Nr. 7710 erhältlich.

Datum: ....

DEMAND PAGING

WAS IST DEMAND PAGING ?

Demand Paging bedeutet: alles (!), was zum Programm gehört (also alle Befehle, Konstanten und Variablen), wird automatisch dann in den Kernspeicher gebracht, wenn man es benötigt, und wird wieder aus dem Kernspeicher entfernt, sobald es nicht mehr gebraucht wird. Auf diese Weise wird für andere Programmteile Platz gemacht.
Alle diese Lade- und Verdrängungsvorgänge finden in Einheiten von einer Seite (= 1024 Worte) statt, um den Kernspeicher optimal auszunutzen und überflüssige Transporte zu sparen.

WARUM BRAUCHT MAN DEMAND PAGING ?

Es gibt zwei unterschiedliche Gründe dafür, Demand Paging zu benutzen:

1. Man möchte mit sehr großen Feldern arbeiten, die nicht mehr in den physikalisch zur Verfügung stehenden Kernspeicher passen (z.B. bei der Bearbeitung großer Matrizen).
2. Man möchte den Kernspeicherbedarf (KSB) eines Programmes reduzieren, um
   a) auch tagsüber das Programm rechnen lassen zu können (KSB-Beschränkungen),
   b) die Liegezeit seiner Aufträge im Rechner zu verkürzen (Verdrängungen wegen Kernspeicherengpässen im System).

WIE BENUTZT MAN DEMAND PAGING AM TR 440 ?

Um die Vorteile des Demand Paging zu nutzen, muß der Benutzer nur ein

□BIBANMELDE, PAGING, LFD

vor dem MØNTIERE-Kommando geben.
Danach läuft - o Wunder - alles wie von selbst, ohne daß am Programm etwas geändert werden muß. Zu beachten ist allerdings, daß alle FØRTRAN- und CØBØL-Programmteile mit VARIANTE = GR übersetzt werden müssen.

WIE FUNKTIONIERT DEMAND PAGING AM TR 440 ?

Man stelle sich einen 1920 K Worte (d.h. ca. 11 Megabyte) großen virtuellen Kernspeicher vor (auch Adreßraum genannt), der jedem Auftrag zur Verfügung steht. Da am TR440 der Kernspeicher 256 K groß ist (wovon wiederum jedem Benutzer nur ein Teil zur Verfügung steht), können bei großen Programmen nicht immer alle Programmteile im Kernspeicher liegen; diejenigen Programmteile, die gerade nicht im Kernspeicher liegen, müssen solange an einem anderen Ort "aufbewahrt" werden, um sie, wenn sie benötigt werden, in den realen Kernspeicher zu laden. Diese Teile liegen auf der Platte. Dort wird bei Programmbeginn ein der Kernspeicheranforderung entsprechender Bereich angelegt und eine eindeutige Zuordnung zwischen einem Wort auf der Platte und seinem zugehörigen Platz im Kernspeicher hergestellt. Wird nun ein Wort vom Programm angesprochen, das noch nicht im Kernspeicher liegt, so wird eine passende 1 K große Umgebung des Wortes auf der Platte gesucht und diese als ein Stück (Seitengebiet genannt) in den Kernspeicher transportiert. Vorher wird dafür gesorgt, daß dort genügend Platz vorhanden ist, indem eventuell ein anderes Seitengebiet vom Kernspeicher auf die Platte verlagert wird.

WIE KLEIN KRIEGT MAN SEINE PROGRAMME ?

Theoretisch können alle Programme bei Benutzung von Demand Paging mit einem Kernspeicherbedarf von 10 bis 15 K laufen - allerdings brauchen sie dann eventuell beliebig viel Rechenzeit.
Praktisch muß man zuerst dafür sorgen, daß das Programm beim Start überhaupt vom System geladen werden kann (vergl. Bemerkung über Laufzeitgebiete auf Seite 4).
Dazu sollte man möglichst viele Programmteile durch Angabe von Vorrangnummern als zuladbar deklarieren (Spezifikation TRANSFER im UEBERSETZE- bzw. MØNTIERE-Kommando).
Der Kernspeicherbedarf bei Programmstart berechnet sich wie folgt:
Man montiere das Programm mit PRØTØKØLL = -STD-.
Dabei erhält man vom Montierer eine Angabe über die Größe des residenten Programmteils (Vorsicht: Angabe ist eine sedezimale Zahl). Dazu addiere man die sedezimale Zahl B. Als Ergebnis erhält man den minimalen Kernspeicherbedarf, der auf jeden Fall zum Starten des Programms benötigt wird.
Außer

BØ&PAGING
S&CC
BCPL&R
ALGOL68-Rahmen
eigenes Hauptprogramm

können alle Programmteile als zuladbar deklariert werden.

WIE KOMMT MAN ZU SEHR GROSSEN FELDERN ?

Bei der Beschaffung großer Felder gibt es in ALGØL und FØRTRAN einige Unterschiede:

1. In ALGØL 60 kann man wie üblich schreiben (z.B.)

   'INTEGER''ARRAY' F[1:250000];

   dabei dürfen jedoch alle Felder zusammen nicht mehr als 250 K groß sein.
   (Bei nachgewiesenem Bedarf kann eine Programmversion zur Verfügung gestellt werden, die bis zu 1024 K erlaubt.)
2. In FØRTRAN kann man die CØMMØN-Zone FTNFSP benutzen, die verlängerbar ist.
   Beispiel:

   ```
   CØMMØN / FTNFSP / F(10)
   ```

   Hierbei ist auch EQUIVALENCE möglich:

   ```
   LØGICAL*1 B(10)
   EQUIVALENCE (F,B)
   ```

   Verlängert wird die Commonzone durch:

   ```
   CALL FTNFSP (200000)
   ```

   Dadurch wird sie 200000 Worte lang.
   Die maximal mögliche Länge bei der Commonzone FTNFSP beträgt 256 K Worte (für Mehrbedarf siehe 1.).
   VORSICHT: Durch die zu kleine Deklaration des Feldes F funktioniert DYNKØN nicht!
   Soll mit DYNKØN übersetzt werden, so empfiehlt sich ein ALGØL 60 - Hauptprogramm, in dem das Feld kreiert und als Parameter an ein FØRTRAN-Unterprogramm durchgereicht wird.
3. Sowohl in FØRTRAN als auch in ALGØL 60 kann man sich mit Hilfe der Prozedur BØ&FSP zusätzlich zu den sonstigen Möglichkeiten 256 K Worte besorgen (siehe RZ-Berichte BØGØL-TAS, BØ.E2.06 und BØTRAN, BØ.E1.14).

Bei allen in ihrer richtigen Länge deklarierten Feldern (auch wenn sie zuladbar sind) ist zu berücksichtigen, daß die Summe der Längen aller zur selben Vorrangnummer gehörenden Felder bei Programmstart einmal kurzzeitig im zur Verfügung stehenden Kernspeicher Platz haben muß (Laufzeitgebiete müssen beim Laden des Programms zur Vorbesetzung komplett in den Kernspeicher).

WAS KANN MAN TUN, UM SEIN "GEPAGETES" PROGRAMM ZU BESCHLEUNIGEN ?

Da bisher noch kein Algorithmus bekannt ist, der ein Programm mit hellseherischen Fähigkeiten ausstattet, steht auch das Demand-Paging-Programm vor dem schwierigen Problem zu entscheiden,welches Seitengebiet den Kernspeicher zu verlassen hat, wenn Platz für ein anderes benötigt wird. Vereinfacht dargestellt, macht es das wie folgt: es wird dasjenige Seitengebiet aus dem Kernspeicher entfernt, das die geringste Anzahl Zugriffe während des bisherigen Programmlaufs aufweist.
Bei dieser schwierigen Entscheidung kann der Benutzer, der (manchmal) mehr über die Struktur und den Aufbau seines Programms weiß, dem Demand-Paging-Programm helfen:

1. Beim Montieren sollten Unterprogramme, die häufig zusammen benutzt werden, die gleiche Vorrangnummer erhalten (Dadurch werden sie "dicht nebeneinander" montiert).
2. Die einzelnen zuladbaren Segmente sollten nicht zu klein sein, damit möglichst wenig Verschnitt an Seitengrenzen entsteht.
3. An Stellen, an denen der Benutzer weiß, daß bestimmte Vorrangnummern nun "längere Zeit" nicht mehr benötigt werden, sollte er sie durch explizites Entladen aus dem Kernspeicher entfernen (Unterprogramm UNLØAD, siehe Seite 7).

WIE KANN MAN INFORMATIONEN ÜBER DIE VORGÄNGE BEIM DEMAND PAGING ERHALTEN, UM OPTIMIEREN ZU KÖNNEN ?

Am Ende des Programms wird als Kurzinformation grundsätzlich ausgedruckt:

1. Anzahl der Pagefaults, d.h. wie oft eine Zelle angesprochen wurde, die gerade nicht im Kernspeicher war.
2. Anzahl der Zuladungen, d.h. wie oft ein Seitengebiet von der Platte in den Kernspeicher verlagert wurde. Diese Zahl ist immer größer als die Anzahl der Pagefaults, da durch interne Optimierungen bei manchen Pagefaults mehr als ein Seitengebiet verlagert wird.

Ein komplettes Ladeprotokoll kann man durch Aufruf des Unterprogrammes LDPEIN ein- bzw. durch LDPAUS ausschalten.
Das erzeugte Ladeprotokoll sieht so aus:

```
***VERDRAENGT: LO5.3.4 *ZUGELADEN: DO2.7
```

Dabei ist

LO5 bzw. DO2 der Name des Plattengebietes, aus dem das Seitengebiet stammt,

3 bzw. 7 die relative Seitennummer innerhalb des Gebietes und

4 die Vorrangnummer, zu der das Gebiet gehört.

Fehlt die Vorrangnummer, so handelt es sich um ein residentes Gebiet (z.B. DO2.7).
Fehlt die Seitennummer, so war das Plattengebiet im Original nur 1 K groß und es brauchte kein Kopie-Seitengebiet angelegt werden (z.B. DO8..10).
Wird statt VERDRAENGT die Meldung ENTLADEN ausgegeben, so handelte es sich um einen expliziten UNLØAD-Aufruf.

UNTERPROGRAMME ZUR STEUERUNG DES DEMAND PAGING

Alle Unterprogramme zur Steuerung des Demand Paging sind von TAS, ALGØL 60, FØRTRAN und CØBØL her aufrufbar.
Als Parameter sind jeweils nur Integerzahlen vorgesehen; die entsprechenden Parametertypen sind

in TAS : Festkommazahlen
in ALGØL 60: 'INTEGER'
in FØRTRAN : INTEGER*4
in CØBØL : PIC 9(13) CØMP.

Folgende Unterprogramme sind aufrufbar:

1. UNLØAD dient zum expliziten Entladen von Seitengebieten, die zu bestimmten Vorrangnummern gehören.
   Beispiel: Entladen der Vorrangnummern 10 und 11
   in TAS : BA(2,10,11),SFBE(UNLØAD/A)
   ↑ Parameterzahl
   in ALGØL 60: UNLØAD(10,11)
   in FØRTRAN : CALL UNLØAD(10,11)
   in CØBØL : ENTER TAS UNLØAD USING Z10, Z11
2. LDPEIN und LDPAUS dienen zur Steuerung des Ladeprotokolls.
   Sie werden parameterlos aufgerufen.
3. LDSTEU dient zum Steuern des Demand Paging.
   Es kann mit ein oder zwei Parametern aufgerufen werden.
   1. Parameter: MAXKSB
      gibt den maximal zu benutzenden Kernspeicher in K Worten an - auch wenn mehr zur Verfügung steht.
   2. Parameter: KSBRES
      gibt an, wieviel K Kernspeicher auf jeden Fall physikalisch frei bleiben sollen.

   Der Kernspeicherbedarf eines Programmes ergibt sich dann als maximal
   MIN (KSB - KSBRES, MAXKSB).
   (KSB ist der dem Auftrag lt. Vermittlerkommando maximal zur Verfügung stehende Kernspeicher.)

Es kann aus zwei Gründen sinnvoll sein, durch MAXKSB bzw. KSBRES Kernspeicher freizuhalten:

1. Braucht das Programm mehr als 100 K virtuellen Kernspeicher, so arbeitet das Demand Paging schneller, wenn 1 K Reserve zur kurzfristigen Benutzung frei bleibt.
2. Die Liegezeit von Aufträgen im Rechner ist meistens kürzer, wenn sie weniger Kernspeicher brauchen (Verdrängungen wegen Kernspeicherengpässen im System).

Die Eingänge GSF, S&BEECALARM, LØAD etc., die man von S&LØAD her kennt, sind in BØ&PAGING auch enthalten, haben aber keinen Einfluß auf den Programmlauf.

ALLGEMEINE BEMERKUNGEN ZUM DEMAND PAGING

1. Ein "gepagetes" Programm braucht etwas mehr Plattenraum als ein "ungepagetes" : maximal ca. 100 K Platte mehr.
2. Alle Testhilfen des TR 440 sind weiterhin voll benutzbar :
   DYNKØN (siehe jedoch Einschränkung für FØRTRAN), TRACE, Kontrollereignisse, Rückverfolger, Sprach-Dumps, etc.
3. Als zusätzliche Anweisungen bei Kontrollereignissen gibt es
   LDPEIN und LDPAUS
   zum Ein- und Ausschalten der Ladeprotokollierung.

Gliederung:

1. ZWECK DES PROGRAMMS

Das Programm ermöglicht es, das bisherige Ablaufprotokoll zu verändern oder in verschieden aufbereiteten Formen in eine Datei zu schreiben zur weiteren bequemen Verarbeitung.

2. AUFBAU

2.1 Programmiersprache: TAS

2.2 Programmierform: Operator

3. HANDHABUNG

3.2 Aufruf:

a) ¤STARTE,BO&PROTOKOLL,LAUF=<Modus>,DATEI=<Sgnr>-<Zieldatei>

b) ¤PROT,DATEI=<Zieldatei>,MODUS=<Modus>

Es bedeuten:

<Sgnr> ::= beliebige symbolische Gerätenummer
(1 = SGNR = 99)

<Zieldatei> ::= Name einer Datei, in die das Ablaufprotokoll kopiert werden soll.

<Modus> ::= Modusangabe für die geforderte Leistung; wird nur ausgewertet, wenn keine Zieldatei angegeben ist.

Datum: 25.8.1974

Die Leistungen des Operators im einzelnen sind:

A.) Es ist keine Zieldatei angeben:

Die Leistung ist abhängig vom angegebenen MODUS:

a) Keine Modusangabe: Das bisherige Ablaufprotokoll wird als Teilauftrag auf dem Schnelldrucker ausgegeben und anschließend gelöscht. Die Seitenzählung läuft danach jedoch normal weiter (wichtig für die Druckerseitenschranke).

b) Modus=KOP: Das bisherige Ablaufprotokoll wird als Teilauftrag ausgegeben, bleibt aber vollständig erhalten.

c) Bei ¤STARTE,BO&PROTOKOLL,LAUF=DEFINIERE

definiert der Operator das Kommando PROT.

d) Modus=SEITE: Ausdrucken der Seitenbelegung des Ablaufprotokolls, d. h. Ausdrucken der aktuellen Seitennummer und zusätzlich, falls diese nicht mit der Seitenzahl übereinstimmt, die aktuelle Seitenzahl (das kann auftreten, wenn das Ablaufprotokoll mittels BO&PROTOKOLL verändert wurde).

e) Modus=SEITENNUMMER: Es wird lediglich die aktuelle Seitennummer ausgedruckt, ohne Überprüfung der Seitenzahl. Dieser Modus benötigt unter Umständen wesentlich weniger Rechenzeit als der Modus SEITE, da bei letzterem das gesamte Ablaufprotokoll gelesen werden muß.

f) Modus=SPERRE: Sperren des Ablaufprotokolls gegen Bearbeitung durch Datenbasis-SSR's - insbesondere gegen weitere Manipulationen durch BO&PROTOKOLL.

g) Modus=LOESE: Eine solche gesetzte Sperre wird wieder aufgehoben.

h) Modus=SPERREGESAMT: Setzen einer solchen Sperre, die nicht wieder aufgehoben werden kann.

i) Modus=ABMELD: Abmelden des Ablaufprotokolls von der Verarbeitung, danach wird jede weitere Eintragung ins Ablaufprotokoll unterdrückt.

j) Modus=ANMELD: Die Abmeldung des Ablaufprotokolls wird wieder rückgängig gemacht, so daß weitere Eintragungen möglich sind.

k) Modus=LOESCHE: Löschen des Ablaufprotokolls.

l) Modus=LSEITE n: Löschen des Ablaufprotokolls ab Seite n; die Seitennumerierung läuft jedoch normal weiter. Die Anzahl der danach belegten Seiten wird protokolliert.

m) Modus=VSABSCHALTE: Abschalten des automatisch generierten Seitenvorschubes sowie Löschen der Kopfzeile.

n) Modus=VSEINSCHALTE: Erneutes Einschalten des automatisch generierten Seitenvorschubes einschließlich der Kopfzeile.

o) Modus=TANR: Ausliefern der Nummer des Teilauftrages, der als nächstes erzeugt wird.

p) Modus=VERKETTEN: Wird das Ablaufprotokoll in eine Datei geschrieben, so wird es hinter den letzten evtl. schon existierenden Satz der Datei geschrieben.

B.) Es ist eine Zieldatei angegeben:

In diesem Fall darf kein Modus angegeben werden.(außer VERKETTEN).

Das Ablaufprotokoll wird in die Zieldatei kopiert zur weiteren Verarbeitung, wird selbst aber nicht verändert. In welcher Weise kopiert wird, ist abhängig vom Typ der Datei - erlaubt sind Dateien vom Typ SEQ, RAN und RAM:

a) Es handelt sich um eine SEQ-Datei:

Die Datei muß den Satzbau "Ausgabezeichen" haben, also z. B. U8OA.

Das Ablaufprotokoll wird mit gebietsweisem Transport (also extrem schnell) unverändert in die SEQ-Datei kopiert. (Speziell dazu gedacht, um das Ablaufprotokoll als Teilauftrag auf ein bestimmtes Gerät zu leiten, z. B. auf ein Sichtgerät, siehe z. B. Kommando DRUCKE etc.) Ist Modus = VERKETTEN angegeben, so wird auf jeden Fall satzweise geschrieben.

b) Es handelt sich um eine RAM-Datei:

Das Ablaufprotokoll wird satzweise in die RAM-Datei kopiert. Dabei werden Ignores und Zeichen mit einem Zentralcodewert <6 sowie das Zeichen 'DEL' (Zentralcodewert = 255) eleminiert, da diese Zeichen auch vom PAV (Papiervermittler) übergangen werden und somit nicht im Ablaufprotokoll erscheinen.

Ist die Datei vom Satzbau Oktaden (z. B. U8OØ), so werden zusätzlich die Vorschubzeichen in jedem Satz am Satzanfang entfernt, und es wird ein Oktadenzähler im letzten Ganzwort eingefügt. Die Datei kann dann mit Texthaltungskommandos (TKOPIERE etc.) bearbeitet werden.

Die Numerierung der Sätze erfolgt in 10-er Schritten: (10,10).

c) Die Zieldatei ist eine RAN-Datei:

In RAN-Dateien wird das Ablaufprotokoll im wesentlichen so wie in RAM-Dateien kopiert, mit zwei Unterschieden:

1. Die Satz-Numerierung ist (1,1).

2. Vor jedem Satz werden 6 Zeichen eingefügt:

   Die ersten 4 Zeichen enthalten die Anzahl der Zeichen des Satzes (z. B. im I4- oder A4-Format einlesbar in FORTRAN), das 5. und 6. Zeichen sind Leerzeichen. (Diese ersten 6 Zeichen sind natürlich in der Zeichenzahl nicht enthalten!).

   Ist der Satzbau der RAN-Datei ≠ Oktaden (z. B. U8OA), so ist in diesem Fall das Vorschubsteuerzeichen des Originalsatzes des Ablaufprotokolls das 7. Zeichen des Satzes in der RAN-Datei.

Diese Form ist vor allen Dingen für die bequeme Handhabung mittels sprachspezifischer E/A gedacht, um Informationen, die andere Operatoren ins Ablaufprotokoll geschrieben haben, auswerten und weiterverarbeiten zu können.

3.3 Speicherbedarf: 5K Kernspeicher

5. FEHLERBEHANDLUNG

a) Ist das Ablaufprotokoll leer oder nicht vorhanden, so wird dies als Fehler gemeldet.

b) Wird ein unzulässiger Modus angegeben, so wird dies als Fehler moniert, danach wird eine Liste aller möglichen Modi mit ihren Bedeutungen ausgegeben.

1. Zweck des Programms

Das UP ermöglicht das Kreieren einer Datei, die über eine symbolische Gerätenummer identifiziert wird, in ALGOL60.

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmierform: Prozedur

3. Handhabung

3.1. Deklaration: procedure DATEI(X); code;

3.2. Aufruf: DATEI(SGNR,S,DTT,DATTR,D,DL,Z,WZ, EXDKZ, KT,EZ,E,BKZ,FVAR);

Bedeutung der Parameter:

SGNR = symbolische Gerätenummer der Datei

S = 0 keine Scheindatei
= 1 Datei als Scheindatei kreieren

DTT = Dateityp
= 1 SEQ
= 2 RAN
= 3 RAM
= 4 RAS
= 5 PHYS

DATTR = Datenträger
= 0 Kernspeicher
= 1 Platte
= 2 Trommel
= 3 verboten!
= 4 Wechselplatte
= 5 LFD

Datum: 1974

D,Z = Interpretation von DL bzw. WZ

= 0 noch nicht definiert
= 1 maximale Anzahl
= 2 genaue Anzahl
= 3 ungefähre Anzahl

DL = Dateilänge, Anzahl der Sätze in einer Datei

WZ = Satzlänge, Anzahl der Ganzworte in einem Satz

EXDKZ = 0 oder

= EXDKZ des Wechselplattenturmes (Algol-String)

KT = Koordinationstyp:

= 1 freie Datei (nur bei Phys-Datei)
= 2 Gemeinschaftsdatei
= 3 Privatdatei (nur sinnvoll bei LFD)

EZ = Anzahl der Satzelemente

E = Elementetyp

= 0 nicht definiert
= 1 Oktaden
= 2 Ganzwörter mit Typenkennung
= 3 Viertelwörter mit Typenkennung
= 4 Ausgabezeichen
= 5 Satzweise verschieden, Ganzwörter oder Oktaden
= 6 Satzweise verschieden, Viertelwörter o. Oktaden

BKZ = 0 oder

= BKZ bei LFD bzw. DMK bei Wechselplatte (Algol-String)

FVAR = Variable, auf der zurückgemeldet wird:

-1, wenn SGNR unzulässig,

sonst: der SSR-Fehlerschlüssel, der beim Kreationsversuch gemeldet wurde.

Beispiel: (Dateikreation wie bei TDEKLARIERE)

```
DATEI(1,0,3,1,3,1,3,13,0,2,80,1,0,7);
      ↑
   1=SGNR in Starte-Kommando
```

≡ Kommando: ¤DATEI, <Dateiname> , RAM-G,U1,U80Ø,P

1. ZWECK DES PROGRAMMS

Es können durch Aufruf des UP's Dateien auf beliebigen Medien als Teilaufträge ausgegeben werden.

2. AUFBAU

2.1. Programmiersprache: TAS

2.2. Programmform: FUNKTIONSPROZEDUR oder PROZEDUR

3. HANDHABUNG

3.1. Deklaration: a) in ALGOL: integer procedure DRUCKE(x); code;

b) in FORTRAN: INTEGER DRUCKE
ALGOL EXTERNAL DRUCKE

3.2. Aufruf: a) I:= DRUCKE (SGNR,ZW,KB,SNR,TYP,GNR,COD, MKZ,SPD,FEHLERLABEL);

SGNR ist die symbolische Gerätenummer, die der auszugebenden Datei zugeordnet wurde.

ZW,KB,SNR,TYP,GNR,COD,MKZ,SPD sind integer-Größen, die in den Versorgungsblock des SSR 253 40 eingespeichert werden (Siehe Handbuch BS3 Systemdienste). Die Standardfälle für diese Größen sind:

ZW Zahl der Wiederholungen (=0 ⇐ einmal)

KB = 0 Ausgabe im O-Format
1 Ausgabe im A-Format
2 Ausgabe im W-Format

SNR = 0 Stationsnummer undefiniert
≠ 0 Stationsnummer

TYP =	Gerätetyp, auf dem die Ausgabe erfolgen soll
0 = '00'	Drucker
1 = '01'	Kartenstanzer
2 = '02'	8-Spur-Streifenstanzer
4 = '04'	Plotter am TR 440
13 = '0D'	Plotter (Gerber- Zeichentisch)
16 = '10'	Fernschreiber
17 = '11'	Fernschreiber
18 = '12'	Sichtgerät
19 = '13'	Sichtgerät
20 = '14'	Kontroll-Schreibmaschine

GNR = Gerätenummer

=0 undefiniert

≠0 Nummer des Gerätes

COD = Gerätecode

Drucker:

1 kleiner Zeichenvorrat

2 großer Zeichenvorrat

3 beliebig

Kartenstanzer:

0 binär

1 KC1

2 KC2

3 KC3

Streifenstanzer:

0 binär

1 SC1

2 SC2

MKZ = Materialkennzeichen

(=0 undefiniert)

SPD =	Steuerparameterdefinition
= 0	es handel sich um einen Ausgabeauftrag
≠ 0	Steuerparameter für das Ablaufprotokoll werden undefiniert
= 1	ZW für das Ablaufprotokoll wird definiert
= 2	Geräteangaben SNR, TYP, GNR, COD und MKZ werden für das Ablaufprotokoll definiert
= 3	ZW, SNR, TYP, GNR, COD und MKZ werden für das Ablaufprotokoll definiert.

Der Funktionswert ist die Nummer des Teilauftrages. Bei irgendeinem auftretenden Fehler wird auf die Marke FEHLERLABEL gesprungen.

Beispiele:

1. I:=DRUCKE(15,0,1,0,0,0,3,0,0,LAB);
   Die Datei mit der Gerätenummer 15 (Sequentielle A-Datei) wird einmal als Teilauftrag auf dem Schnelldrucker ausgegeben. Die Datei ist anschließend nicht mehr vorhanden.

2. I:=DRUCKE(15,8,2,0,1,0,0,0,0,LAB);
   Die Datei mit Gerätenummer 15 wird 9-mal auf Karten binär gestanzt (Datei bleibt erhalten).

3. I:=DRUCKE(15,0,0,1,18,48,3,0,0,4711) { 18≙Sichtgerät}
   Die Datei mit der Gerätenummer 15 wird einmal auf dem Sichtgerät I48-1 ausgegeben.

b) Aufruf in FORTRAN entsprechend, nur das Fehlerlabel muß bei Benutzung als Funktionsprozedur aus syntaktischen Gründen entfallen (siehe 5. Fehlerbehandlung).

3.3. Speicherbedarf:

176 Befehle
3 Ganzworte Konstanten
6 Ganzworte Arbeitsspeicher

4. ARBEITSWEISE

4.1. Verfahren:
Die Datei wird mit dem SSR253 40 ausgegeben. Ist ZW > 0, so bleibt die Datei erhalten, sonst ist sie anschließend gelöscht.

4.2. Gültigkeitsbereich:
Es sind nur SEQ-Dateien (nicht LF- oder WSP-Dateien) zulässig. Die Dateien müssen von der Bearbeitung abgemeldet sein.

5. Fehlerbehandlung

Bei irgendeinem auftretenden Fehler wird auf das angegebene Fehlerlabel gesprungen.

Ist kein Fehlerlabel angegeben (in FORTRAN darf bei Funktionsprozeduren kein Label als Parameter verwendet werden), so wird als Funktionswert ein Fehlerschlüssel übergeben:

=O bei einem formalen Fehler - wenn z. B. der symbolischen Gerätenummer im Starte-Kommando keine Datei zugeordnet wurd.

= -k bei einem SSR-Fehler während des SSR 253 4O. Dabei ist k der SSR-Fehlerschlüssel (siehe Handbuch BS3-Systemdienste), und zwar die rechten 16 Bits.

DIALOG

Spezifikation:

Spez.-Wert:

Keine Spezifikationen

Kommando für das Programmiersystem	anlagenspezifische Voreinstellung:

Einschränkung:

Wirkung:

Mit Hilfe dieses Kommandos läßt sich ein Dialog zwischen zwei Gesprächen herstellen, der unabhängig von evtl. parallel ablaufenden anderen solchen Dialogen durchgeführt werden kann. Die eindeutige Identifizierung der Dialogpartner wird dabei durch die Bezeichnung der jeweiligen Terminals hergestellt.

Zu diesem Zweck fragt der Operator BO&KOMVOR, der die Leistungen des Kommandos DIALOG erbringt, zu Beginn nach der Bezeichnung des Terminals, an dem der gewünschte Dialogpartner arbeitet.

Diese Gerätebezeichnung muß in der Form

⟨Typ⟩⟨GNR⟩-⟨SNR⟩

angegeben werden, also z. B.: F75-1
oder I96-1
oder X80-1 etc.
(F für Fernschreiber, I für Sichtgerät, X für Wählgerät).

Liegt von diesem Gerät schon eine Dialogsendung vor, so wird diese ausgedruckt und dann nach der Antwort gefragt.

Liegt noch keine Sendung vor, so wird zunächst eine Benachrichtigung an das angegebene Terminal gesendet, daß der Wunsch nach einem Dialog besteht - unter Spezifizierung des Absender-Terminals sowie des Namens des Absenders (aus dem BEN-String entnommen) und danach der erste Dialog-Text angefragt.

Auf eine Anfrage nach Information können bis zu 767 Zeichen eingegeben werden, die beim Dialogpartner ausgedruckt werden sollen.

Danach wird auf eine entsprechende (Antwort-)Sendung gewartet.

Der Dialog wird beendet durch eine leere Eingabe.

Befindet sich ein Dialogpartner im Zustand "Warten auf Antwort", und dauert es dem Benutzer entweder zu lange oder ist keine Antwort mehr zu erwarten, so kann er durch Eingabe von "XAN." und (nach der Meldung des Abwicklers "ABW:") durch anschließendes "HALT." den Operator veranlassen, den Wartezustand zu verlassen und wieder nach Information zu fragen.

Dadurch ist es möglich, einen solchen Dialog jederzeit zu beenden, indem eine Anfrage nach Information mit leerer Eingabe quittiert wird.

Wird die Anfrage nach dem Terminal des Dialogpartners mit einem Fragezeichen beantwortet, so wird eine Liste aller Gespraeche einschließlich der Gerätebezeichnungen ausgegeben und die Frage anschließend wiederholt.

DTSENDE

Spezifikation:

1	DATEI	Angabe der zu versendenden Datei
2	KENNZEICHEN	Kennzeichnung der Sendung
3	VERFALLSZEIT	Verfallszeit der Sendung

Kommando für das Programmiersystem

anlagenspezifische Voreinstellung:

Einschränkung:

Wirkung:

Eine Datei wird in ein intern kreiertes Gebiet transportiert und dieses Gebiet dann als Sendung mit dem ABS-NAMEN KENNZEICHEN abgeschickt. Wird eine solche Sendung mit dem Kommando EMPFANGE übernommen, so wird diese Datei identisch beim Empfänger kreiert und steht dann dem Empfänger zur Bearbeitung zur Verfügung.
Die Datei selbst wird beim Absender in keinem Fall verändert, sondern es wird eine Kopie beim Empfänger erzeugt. (Beschreibung der Leistungen siehe auch unter Kommando SENDE bzw. EMPFANGE).

formal :

Beispiel :

DATEI

Spez.-Wert:

datei	Name der Datei, die gesendet werden soll - die Datei liegt in der Standarddatenbasis
db. datei	Die Datei datei in der Datenbasis db soll gesendet werden und beim Empfänger auch in der Datenbasis db kreiert werden.

obligate Spezifikation zum Komm. DTSENDE

anlagenspezifische Vorbelegung: "undefiniert"

Einschränkung:

Die Sätze der Datei dürfen nicht länger als 1022 Ganzworte sein.
Eine RAS-Datei muß den Satzschlüssel im Satz haben.

Wirkung:

Die angegebene Datei wird einschließlich Kenndaten und Satzmarken in ein Gebiet transportiert und dieses Gebiet dann gesendet (SSR 5 8).
Erlaubt sind SEQ-,RAN-,RAM-,RAS- und PHYS-Datei (siehe jedoch unter Einschränkung).
Dateien werden immer als Gebietssendung verschickt, und es wird auf deren Rücksendung gewartet.

formal :

Beispiel :

KENNZEICHEN — Kennzeichnung der Sendung

Spez.-Wert:

Normalstring	Zeichenfolge, durch die die Sendung identifiziert wird.

obligate Spezifikation zum Kommando DTSENDE | anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Es werden maximal die ersten 12 Zeichen ausgewertet

Wirkung:

Durch die angegebene Zeichenfolge wird die Sendung mit einem Namen versehen, durch den sie beim Empfangen (siehe Kommando EMPFANGE) eindeutig bezeichnet werden kann, wenn keine andere Sendung mit dem gleichen Kennzeichen vorhanden ist.
Das Kennzeichen sollte deshalb so gewählt werden, daß die Wahrscheinlichkeit gering ist, daß jemand anders das selbe genommen hat.
(KENNZEICHEN wird beim SSR 5 8 als ABS-NAME eingesetzt).

formal:

<Wertzuwsg. KENNZEICHEN>::=[KENNZEICHEN =]<Normalstring>

<Normalstring> ::= siehe Kommandohandbuch

Beispiel:

..., KENNZEICHEN = (KE1*KE2),...

VERFALLSZEIT

Spez.-Wert

"undefiniert"	Die Sendung existiert so lange wie die Warteschlange (ca. 10 Tage)
tt hh mm	tt = Anzahl der Tage hh = Anzahl der Stunden mm = Anzahl der Minuten

	Vorbelegung: "undefiniert"

Einschränkung

wenn VERFALLSZEIT ≠ "undefiniert" ist, wird mindestens 1 Minute eingesetzt.

Wirkung

Die VERFALLSZEIT gibt an, wie lange die Sendung maximal existieren soll.
Außerdem wartet der absendende Auftrag maximal so lange, wie VERFALLSZEIT angibt, auf die Rücksendung des Empfängers, um die Speicherberechtigung zurückzuerhalten. Trifft nach VERFALLSZEIT keine Rücksendung ein, so wird der Fehler

"+++++ Sendung wurde nicht übernommen"

gemeldet und die Sendung wieder gelöscht.

formal:

<Wertzuwsg. VERFALLSZEIT>::= { [[<T>] <H>] <M> | — }

<T> ::= [t] t
<H> ::= [h] h
<M> ::= [m] m

Beispiel:

```
..., VERF.=    10
 10 Minuten Verfallszeit
..., V.   =   103
  1 Stunde und 3 Minuten Verfallszeit
..., V.   = 30520
  3 Tage, 5 Stunden und 20 Minuten Verfallszeit
```

EMPFANGE

Spezifikation:

1	KENNZEICHEN	Kennzeichnung der Sendung
2	WARTEZEIT	Angabe der maximalen Zeit, die auf die Sendung gewartet werden soll

Kommando für das Programmiersystem	anlagenspezifische Voreinstellung:

Einschränkung:

Wirkung:

Es wird (evtl. nach einer gewissen Wartezeit) die durch KENNZEICHEN spezifizierte Sendung übernommen und ausgewertet. Folgende Sendungen sind möglich:

1. Die Sendung wurde durch das Kommando SENDE erzeugt. In diesem Falle enthält sie Kommandos, die beim Auswerten durch Start des Entschlüßlers ausgeführt werden.

2. Die Sendung wurde durch das Kommando DTSENDE erzeugt. In diesem Fall enthält sie den Inhalt einer Datei, die dann kreiert wird. Existiert die zugehörige Datenbasis nicht, so wird sie kreiert. Existiert bereits eine gleichnamige Datei, so geschieht folgendes:

   a) die existierende Datei ist eine eingeschleuste Datei: dann wird diese vorher abgemeldet.

   b) sonst wird die existierende Datei vorher gelöscht.

   Existiert die Datenbasis beim Empfänger nicht, so wird sie implizit kreiert.

3. Die Sendung wurde durch das Kommando SENDE erzeugt, aber mit der Spezifikationsangabe MAL=-STD-. In diesem Fall besteht die Sendung aus einer Textfolge, die beim Empfänger ausgedruckt wird (ins Ablaufprotokoll und auf

dem Terminal).
Paßte die Textfolge in eine Seite, so handelt es sich nur um eine Kernspeichersendung, die nicht zurückgesandt zu werden braucht, da der Absender nicht auf die Rücksendung wartet und die Sendung auf Kosten der Warteschlange gesendet wurde. Andernfalls handelt es sich um eine Gebietssendung, die nach der Ausgabe sofort zurückgeschickt wird, um dem Absender seine Speicherberechtigung zurückzubuchen (Gebietssendungen werden immer zu Lasten des Absenders geschickt).

4. Die Sendung wurde durch das Kommando SENDE erzeugt, aber mit der Spezifikationsangabe QUELLE=-STD-. Dann besteht die Sendung nur aus dem Sendungskopf. Diese Form dient dazu, zu Koordinationszwecken auf eine solche Sendung warten zu können, ohne daß eine Informationsübergabe nötig wäre.

5. Beim Senden ist ein Fehler aufgetreten (Speichermangel etc.):
Dann besteht die Sendung nur aus einem Fehlerschlüssel, der eine Fehlermeldung beim Empfang spezifiziert (KSP-Sendung).

Handelt es sich um eine Kernspeicher-Sendung oder nur um einen Sendungskopf, so wird die Sendung automatisch beim Übernehmen gelöscht.
Andernfalls wird die Sendung nach der Auswertung zurückgeschickt, denn der absendende Auftrag wartet dann auf die Rücksendung, um seine Speicherberechtigung zurückzuerhalten.
Bei 1. wird das übernommene Gebiet zuerst kopiert, dann zurückgeschickt und danach erst der Entschlüßler gestartet.
Bei 2. wird das Gebiet erst nach der vollständigen Erstellung der Datei zurückgeschickt.
Tritt beim Empfang irgendein Fehler auf, so wird eine Gebietssendung auf jeden Fall noch zurückgeschickt.
Ein Auftrag kann eine zweite Sendung erst abschicken, nachdem die vorige Sendung übernommen und zurückgeschickt wurde oder die Verfallszeit überschritten und die Sendung deshalb wieder gelöscht wurde. Der Typ der Sendung wird am ABS-FKZ erkannt (siehe SSR 5 8 etc).

KENNZEICHEN Kennzeichen der Sendung, die übernommen werden soll

Spez.-Wert:

Normalstring	Zeichenfolge, durch die eine Sendung identifiziert wird.

obligate Spezifikation zum Komm. EMPFANGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Es werden maximal die ersten 12 Zeichen ausgewertet

Wirkung:

Durch die angegebene Zeichenfolge wird die Sendung gekennzeichnet, die übernommen werden soll bzw. auf die gewartet werden soll.
Hier muß dieselbe Zeichenfolge angegeben werden wie beim KENNZEICHEN des dazugehörigen SENDE-Kommandos.

EMPFANGE KENNZEICHEN

formal:

<Wertzuwsg: KENNZEICHEN>::=[KENNZEICHEN=]<Normalstring>

<Normalstring> ::= siehe Kommandohandbuch

Beispiel:

..., KENNZEICHEN= (KE1*KE2),...

WARTEZEIT

Spezifikation:

Spez.-Wert:

"undefiniert" :	Es soll auf keinen Fall gewartet werden
tthhmm. ss	Angabe der maximalen Wartezeit auf die spezifizierte Sendung: tt = Anzahl der Tage hh = Anzahl der Stunden mm = Anzahl der Minuten ss = Anzahl der Sekunden

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Ist WARTEZEIT = "undefiniert", so wird nur nachgesehen, ob die Sendung vorhanden ist: wenn ja, wird sie übernommen, andernfalls wird der Fehler:"+++++ Sendung nicht vorhanden" gemeldet.

Ist unter VERFALLSZEIT eine Zeitangabe gemacht, so wird maximal so lange wie angegeben auf das Eintreffen der Sendung gewartet. Trifft sie innerhalb dieser Zeit ein oder war sie vorher schon vorhanden, so wird sie übernommen, andernfalls wird nach Ablauf der Wartezeit der Fehler "+++++ Wartezeit überschritten" gemeldet.

EMPFANGE WARTEZEIT

Format:

siehe Kommando SENDE

Beispiel

..., W.=.15
Es wird max. 15 Sekunden gewartet

..., WARTEZEIT=1324.47
Es wird max. 13 Stunden, 24 Minuten und 47 Sekunden gewartet

ERZEUGE

ERZEUGE	Manipulationen mit Elementen der Kommandosprache

Spezifikation:

1	ZIEL	: Gewünschte Dienstleistung oder zu besetzender interner Name
2	QUELLE	: Ausgangstext
3	TEILWERTE	: Texte zur Modifikation der QUELLE
4	KRITERIUM	: Auswahl- oder Abbruchkriterien
5	MODUS	: Angabe der gewünschten Interpretation der Spezifikationen TEILWERTE und KRITERIUM
6	MAL	: Auszeichnung eines Zeichens als Mal
7	PROTOKOLL	: Steuerung der Protokollierung

Kommando für das Programmiersystem	

Einschränkung:

Wirkung:

Elemente der Kommandosprache - Normal- und Fremdstrings - können auf vielfältige Art verarbeitet werden.

Möglich sind z. B.

- Verkettung mehrerer Spezifikationswerte zu einer Zeichenfolge,
- Auswahl bestimmter Teilwerte aus einer Liste,
- Interpretation von (Teil-)Zeichenfolgen als Formeln bzw. arithmetische Anweisungen,
- Bestimmung der Anzahl von Teilwerten einer Liste.

Die Ergebnisse werden wie unter ZIEL angegeben einem internen Namen zur Weiterverarbeitung zugewiesen oder sie bewirken eine Dienstleistung (Ausführung oder Definition von Kommandos).

Die Bearbeitung geht von dem unter QUELLE angegebenen Text aus, der je nach MODUS mit Teilwerten von TEILWERTE modifiziert wird. In Abhängigkeit von MODUS werden die Angaben unter KRITERIUM als Indizes oder Abbruchkriterien aufgefaßt.

Das als MAL definierte Zeichen gestattet besondere Bearbeitung beliebiger Stellen des Textes.

formal:

⟨ERZEUGE-Kommando⟩ ::= ◊ERZEUGE [, [⟨Spezifikationsname⟩ =] ⟨Spezifikationswert⟩] ∞

⟨Spezifikationsname⟩ ::= ZIEL | QUELLE | TEILWERTE | KRITERIUM | MØDUS | MAL | PRØTØKØLL

Beispiel:

```
/STD.,C#TEINTR.,A,INF.=/
DREI CHINESEN MIT DEM KONTRABASS
SASSEN AUS DER STRASSE UND ERZAEHLTEN SICH WAS,
DA KAM DIE POLIZEI UND SAGT': 'WAS IST DENN DAS?'
DREI CHINESEN MIT DEM KONTRABASS!
#*SLOCKUNG(CODEW.)=1
/ERZ.,KOPW.,/CODEW.,C,A,1,/ART=2,A=*!1,F=*!1,I=*!1,C=*!1,U=*!1,
TRFZE1='AF','31',*AD4U.,C,KO'DN#/,MOD.=SCHL.,TEI.=*'E'I'O'U
,MAL=*
#.
KREIERT: C(1111.11)

DIE ZIELDATEI A(1111.11) WURDE ERSTELLT

ENDE TEINTRAGE (6.08)  1.13

FOLGENDE DB-DATEI(EN) WURDE(N) VERAENDERT:
DATENBASIS: *STDD2, DATEI: C

ENDE BDASCODE2 (3.03)  1.17

DREI CHANASAN MAT DAM KANTRABASS
SASSAN AAF DAR STRASSA AND ARZAAHLTAN SACH WAS,
DA KAM DAA PALAZAA AND SAGT': 'WAS AST DANN DAS?'
DREI CHANASAN MAT DAM KANTRABASS!

ENDE DAUSGABE (13.16)  1.09

FOLGENDE DB-DATEI(EN) WURDE(N) VERAENDERT:
DATENBASIS: *STDD2, DATEI: C

ENDE BDASCODE2 (3.03)  1.15

DREE CHENESSEN MET DEM KENTREBESS
SESSEN EEF DER STREESSE END ERZEEHLTEN SECH WES,
DE KEM DEE PELEZEE END SEGT': 'WES EST DENN DES?'
DREE CHENESSEN MET DEM KENTREBESS!

ENDE DAUS      13.16)
```

ERZEUGE

ZIEL

(1)

	ZIEL	Angabe von internen Namen oder Tätigkeiten

Spez.-Wert:

TEXT	:	Das Ergebnis wird ausgedruckt
n	:	Das Ergebnis wird dem internen Namen *n zugewiesen.
KOMMANDO	:	Das Ergebnis ist eine Kommandofolge, die ausgeführt wird.
BEREICH	:	Das Ergebnis wird als die Bezeichnung eines Problemkomplexes aufgefaßt, zu dem Kommandos definiert oder ausgeführt werden sollen.

Mehrere Angaben sind durch Apostroph zu trennen.

Obligate Spezifikation z. Kdo. ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Die im Falle BEREICH zulässigen Problemkomplexe sind rechenzentrumsspezifisch.

Wirkung:

In den Modi INDIZIERUNG und SCHLEIFE entstehen wie unter MODUS und QUELLE beschrieben ein oder mehrere Ergebnisse, die der Reihe nach den Teilwerten von Ziel zugeordnet werden. Lautet der zugeordnete Teilwert TEXT, so wird das Ergebnis ausgedruckt; ist er eine natürliche Zahl, so wird das Ergebnis dem dadurch bezeichneten internen Namen zugewiesen. Im Falle KOMMANDO wird das Ergebnis als Kommandofolge interpretiert, im Falle Bereich muß es die Bezeichnung eines zu aktivierenden Problemkomplexes sein.

Entstehen mehr Ergebnisse, als Ziele angegeben wurden, so wird das letzte definierte Ziel allen weiteren Ergebnissen zugeordnet.

Im Modus ANZAHL sind nur interne Namen als Ziele zulässig.

1 OKT. 1974

formal:

<Wertzuweisung ZIEL> ::= [ZIEL=] <Zielangabe> ['<Zielangabe>]$^{\infty}$

<Zielangabe> ::= <Adresse eines internen Namens> | KØMMANDØ | TEXT | BEREICH

<Adresse eines internen Namens> ::= <Ziffer ≠ o>[<Ziffer>]^{5}

Die Angaben KØMMANDØ und BEREICH dürfen wie Tätigkeitsnamen abgekürzt werden.

Beispiel:

◊ERZ.,ZIEL=KØMM.'BER.'13,,LFI.'ABK.'AFFE,MØD.=SCHL.

Bewirkt wird

1. Ausführung des Kommandos ◊LFI. ,
2. Definition der Kommandos aus dem Bereich "Abkürzungen" ,
3. Zuweisung *13=AFFE

ERZEUGE
QUELLE

QUELLE	Angabe des Ausgangstextes zur Verarbeitung	2

Spez.-Wert:

Zulässig ist eine beliebige Folge von Normal- oder Fremdstrings, die durch Apostroph voneinander getrennt werden.

"undefiniert" : Der Ausgangstext enthält keine Zeichen.

Optionale Spez. zum Kommando ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Alle Teilwerte werden in der angegebenen Reihenfolge (ohne die trennenden Apostrophe und ohne Fremdstringbegrenzer) zu einem Text verkettet.
Durch <Mal>!<Zeichen> oder <Zeichen><Mal>;
wird ein Platzhalter definiert, der
je nach MODUS und evtl. KRITERIUM durch einen Teilwert von TEILWERTE ersetzt wird. Die Quelle kann mehrere Platzhalter enthalten, die, wenn die <Zeichen> verschieden sind, durch verschiedene, sonst durch gleiche Teilwerte ersetzt werden.

Alle Male, denen kein! oder ; folgt, werden wie unter MAL beschrieben interpretiert.

Enthält die Quelle keine Platzhalter, so wird einer als am Ende stehend angenommen.

6. OKT. 1974

formal:

⟨Wertzuweisung QUELLE⟩ ::= [QUELLE=] { ⟨Spezifikationswert⟩ [' ⟨Spezifikationswert⟩] }

⟨Spezifikationswert⟩ ::= ⟨Normalstring⟩ | /⟨Fremdstring⟩◊/

Beispiel:

Q.=/MB(◊/'900013'/)◊/,...

TEILWERTE	Texte zur Modifikation der QUELLE	③

Spez.-Wert:

Zulässig ist eine beliebige Folge von Normal- oder Fremdstrings, die durch Apostroph voneinander getrennt werden.
Einzelne leere Teilwerte müssen als leere Fremdstrings dargestellt werden, da "undefiniert" als Teilwert verboten ist.

"undefiniert" : Die Liste der Teilwerte ist leer.

Optionale Spez. zum Kommando ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Im Gegensatz zur QUELLE werden die TEILWERTE nicht verkettet; sie bilden vielmehr eine (eindimensionale) Liste, aus der einzelne Elemente ausgewählt werden.

Enthält die Liste zu wenig Elemente, so wird an Stelle eines nicht vorhandenen der letzte Teilwert genommen; TEILWERTE="undefiniert" wird wie TEILWERTE="leerer String" behandelt.

1. OKT. 1974

formal:

⟨Wertzuweisung TEILWERTE⟩ ::= [TEILWERTE=] { ⟨Spezifikationswert⟩ [' ⟨Spezifikationswert⟩]∞ | - }

⟨Spezifikationswert⟩ ::= ⟨Normalstring⟩ | /⟨Fremdstring⟩◊/

Beispiel:

◊ERZ.,KØMM.,/EINSCHL.,KAT.◊/,TEILW.=A'B'C,MØD.=SCHL.

Es werden nacheinander die Dateien

KAT.A
KAT.B
KAT.C

eingeschleust.

KRITERIUM	Auswahl- oder Abbruchkriterien

Spez.-Wert:

n : Index im Falle INDIZIERUNG,
Wiederholungsfaktor im Falle SCHLEIFE

ERFOLG, FEHLER } : Abbruchkriterien für SCHLEIFE

"undefiniert" bedeutet soviel wie 1 .

Mehrere Angaben sind durch Apostroph zu trennen.

Optionale Spezifikation z. Kdo. ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Die Bedeutung richtet sich nach dem MODUS:

MODUS=INDIZIERUNG:

Als Kriterien sind nur natürliche Zahlen zulässig, die als Indizes eine Auswahl von Teilwerten der Spezifikation TEILWERTE bewirken.

MODUS=SCHLEIFE:

Abhängig von der Anzahl der Platzhalter und TEILWERTE entstehen mehrere Ergebnisse, denen der Reihe nach je ein Teilwert von KRITERIUM und einer von ZIEL zugeordnet werden.
Bezeichnet der betreffende Teilwert von ZIEL einen internen Namen, so ist die entsprechende Angabe zu KRITERIUM bedeutungslos; in den Fällen KOMMANDO und BEREICH steuert sie die Anzahl der Ausführungen:

n : Die Ausführung erfolgt n mal.
ERFOLG : Die Ausführung wird so lange wiederholt, bis sie erfolgreich gelingt.
FEHLER : Die Ausführung wird solange wiederholt, wie sie fehlerfrei gelingt.

MODUS=ANZAHL:

Die Angabe zu KRITERIUM ist bedeutungslos.

formal:

⟨Wertzuweisung KRITERIUM⟩ ::= [KRITERIUM=] { ⟨Kriterium⟩ ['⟨Kriterium⟩]∞ | - }

⟨Kriterium⟩ ::= ⟨Zahl > o⟩ | FEHLER | ERFØLG

Die Angaben FEHLER und ERFØLG dürfen wie Tätigkeitsnamen abgekürzt werden.

Beispiel:

```
...,MØD.=SCHL.,KRIT.=1'FEHLER'ERF.'2
```

ERZEUGE
MODUS

MODUS	Art der Interpretation von TEILW. und KRIT. (5)

Spez.-Wert:

ANZAHL : Die Anzahl der TEILWERTE wird ermittelt.

INDIZIERUNG : Durch KRITERIUM werden einzelne TEILWERTE zur Modifikation der QUELLE ausgewählt.

SCHLEIFE : Die QUELLE wird der Reihe nach durch alle TEILWERTE modifiziert; KRITERIUM beeinflußt die Bearbeitung jedes Ergebnisses.

"undefiniert" bedeutet soviel wie INDIZIERUNG .

Optionale Spezifikation z. Kdo. ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

ANZAHL:

Die Anzahl der Teilwerte von TEILWERTE wird allen unter ZIEL angegebenen internen Namen zugewiesen.
Von den restlichen Spezifikationen wird lediglich PROTOKOLL ausgewertet.

INDIZIERUNG:

Ist p die Anzahl der (verschiedenen) Platzhalter von QUELLE, so werden die ersten p Teilwerte von KRITERIUM als Indizes aufgefaßt, die die einzusetzenden p TEILWERTE auswählen. Das Ergebnis wird gemäß der ersten ZIEL-Angabe weiterverarbeitet. Wurden bei ZIEL mehr als eine oder bei KRITERIUM mehr als p Angaben gemacht, wird solange analog weiterverfahren, bis alle Angaben ausgewertet sind; dabei wird ggf. von der zuerst erschöpften Spezifikation der letzte Teilwert mehrfach genommen.

SCHLEIFE:

Sei p die Anzahl der Platzhalter und k das Maximum der Anzahlen der Ziele und Kriterien, dann werden in einer Schleife ($1 \leq i \leq k$) die TEILWERTE der Position i bis i+(p-1), also immer p TEILWERTE der Reihe nach, für die Platzhalter eingesetzt. Das Ergebnis wird dem i-ten ZIEL zugeordnet; das i-te KRITERIUM beeinflußt die Ausführung.
Ist die Anzahl der Ziele und Kriterien verschieden, so wird von der zuerst erschöpften Spezifikation der letzte Teilwert mehrfach genommen.

16. OKT. 1974

formal:

⟨Wertzuweisung MØDUS⟩ ::= [MØDUS=] { INDIZIERUNG | SCHLEIFE | ANZAHL }

Die Angaben dürfen wie Tätigkeitsnamen abgekürzt werden.

Beispiel:

◊ERZ.,13,,A'B'C,MØD.=ANZ.

Die Anzahl der Teilwerte (hier: 3) wird dem internen Namen *13 zugewiesen.

◊ERZ.,KØMM.,,A'B,KRIT.=*2,MØD.=IND.

Ist der Wert von *2 gleich 1, wird das Kommando ◊A ausgeführt, ansonsten das Kommando ◊B.

6

MAL	Definition eines Zeichens als Mal

Spez.-Wert:

z : Angabe genau eines Zeichens

/z
/z0/ : Im Normalstring verbotene Zeichen können als Fremdstring angegeben werden.

n : Angabe des Zentralcodewertes des gewünschten Zeichens ($n \geq 16$)

"undefiniert" bedeutet soviel wie 53 (Zeichen FL)

Optionale Spez. zum Kommando ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Wird im Text ein Mal gefunden, so richtet sich der Verlauf der weiteren Interpretation nach dem darauffolgenden Zeichen. Im folgenden sei das Mal durch § dargestellt.

§[und §] wirken als Stringklammern. Innerhalb eines so eingeklammerten Strings werden Male nur erkannt, wenn sie weitere, geschachtelte Stringklammern einleiten. Das äußerste Klammernpaar wird entfernt. Auf diese Weise kann das Ergebnis Male enthalten, die erst bei einer nachfolgenden Verarbeitung wirksam werden.

§+ und §+ wirken als Formelklammern. Die Zeichenfolge §+‹Formel›§+ wird durch das Ergebnis der Interpretation der ‹Formel› ersetzt. Eine Formel kann keine weiteren Formelklammern enthalten.

Gleichbedeutend mit §!‹Zeichen› ist ‹Zeichen›§; .

§! definiert einen Platzhalter der Form §!‹Zeichen› . Mit gleichem ‹Zeichen› benannte Platzhalter werden durch gleiche TEILWERTE ersetzt. Die Auswahl der TEILWERTE richtet sich nach dem MODUS und evtl. KRITERIUM.

Folgt dem Mal eine Ziffer, so wird eine genau dreistellige Zahl n erwartet ($0 \leq n \leq 255$). §n wird durch das Zeichen mit dem Zentralcodewert n ersetzt.

In allen anderen Fällen wird das Mal bei einer nachfolgenden Ausführung des Ergebnisses als Kommandofolge zum Fluchtsymbol.

16. OKT. 1974

ERZEUGE/MAL

formal:

⟨Wertzuweisung MAL⟩ ::= [MAL=] ⟨Zeichen⟩

⟨Zeichen⟩ ::= ⟨Zeichen ≠ *| =| /| ,| (|)| '⟩
/⟨beliebiges Zeichen⟩ [◊/]
⟨natürliche Zahl n, 16 ≤ n ≤ 255⟩

Beispiel:

...,MAL=/)

ERZEUGE
PROTOKOLL

PROTOKOLL	Steuerung der Protokollierung	(7)

Spez.-Wert:

"undefiniert" : Abgesehen von einer kurzen Meldung der MV-Nummer des Operators und der benötigten Rechenzeit werden nur Fehlermeldungen und Warnungen ausgegeben.

&Z : Unterdrückung der Zeitmeldung

A : Ausführliches Protokoll

KØ : Zusätzliche Protokollierung auf Konsole

KW : Keine Ausgabe von Warnungen

(Ø : Objektprotokoll, nur in Fehler- und Testfällen von Interesse)

Mehrere Angaben sind durch Apostroph zu trennen.

Optionale Spez. zum Kommando ERZEUGE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Durch die Angaben wird der Grad der Protokollierung beeinflußt:
Soll sich ERZEUGE völlig stumm verhalten (abgesehen von Warnungen und Fehlermeldungen), so ist PROTOKOLL=&Z anzugeben. Ein ausführliches Protokoll wird dagegen durch Angabe von A angefordert.
Die Ausgabe von Warnungen läßt sich durch KW unterdrücken.

Im Normalfall werden im Gespräch nur Fehlermeldungen und Warnungen auf Konsole ausgegeben; wird ein ausführliches Protokoll auch auf Konsole gewünscht, ist zusätzlich KØ anzugeben.

Eine Angabe von Ø ist nur sinnvoll, wenn das Protokoll zwecks Fehlerverfolgung weitergereicht werden soll.

Fehlermeldungen lassen sich selbstverständlich nicht unterdrücken.

16. OKT. 1974

ERZEUGE/PRØTØKØLL

formal:

⟨Wertzuweisung PRØTØKØLL⟩ ::= [PRØTØKØLL=] { - | ⟨Angabe⟩['⟨Angabe⟩]∞ }

⟨Angabe⟩ ::= &Z | A | KØ | KW | Ø

Beispiel:

```
PRØT.=&Z'KW
```

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Konstruktionsbeschreibung	BØ E1 . 18 K
Programm-Thema: "Stumme" Version von F&STOP	Prog.-Name F&STOP

An alle FTN-Programme wird intern F&STOP anmontiert und bei Auftreten einer STOP- oder PAUSE-Anweisung aufgerufen. Dadurch druckt z. B. jedes FTN-Programm vor der eigentlichen Endemeldung das Wort STOP aus, was oft nicht erwünscht ist.

Die F&STOP-Version der Bibliothek UNIHIP (Träger LFD) unterdrückt die Worte PAUSE und STOP, nicht aber eine etwa angegebene Literalkonstante.

Durch

```
PAUSE '<Text>'
```

läßt sich so leicht ein Text ins Prottokoll absetzen;

```
STOP
```

bewirkt stillschweigendes Beenden des Programmlaufs.

Datum: .....

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung | BØ.E2.07 B

Programm-Thema: Kenndaten (SSR 4 O) für Algol- und Fortranprogramme | Prog.-Name: KENDAT

Gliederung:

1. Zweck des Programms

KENDAT liefert Kenndaten des aktuellen Abschnitts und Operatorlaufes in für Algol- und Fortran-Programme aufbereiteter Form aus. Die Information wird in einen Common-Bereich mit dem Namen KENDAT abgelegt.

2. Aufbau

2.1 Programmiersprache TAS

2.2 Programmform:

procedure	in ALGOL
bzw. SUBROUTINE	in FTN
bzw. rekursive Funktion oder Routine	in BCPL

3. Handhabung

3.1 Deklaration:

a) in Fortran nicht nötig
b) in Algol: procedure KENDAT; code;
c) in BCPL: EXTERNAL KENDAT: B. KENDAT, B. KENNDATEN

3.2 Aufruf:

a) in Fortran: CALL KENDAT
b) in Algol: KENDAT;
c) in BCPL: B.KENDAT()

3.3 Speicherbedarf:

292 Befehle
7 Ganzworte Konstanten
154 Ganzworte Arbeitsspeicher

Der Common-Block, in den die Information abgelegt wird, muß folgenden Aufbau haben:

a) in Fortran:

```
COMMON/KENDAT/NKSB, NBGB, NTSB, NPSB,
NDRS, NSBG, NRZS, NANR, NTYP, NGNR,
NSNR, NKSPMX, NTSPMX, NPSPMX,
```

Datum: - 5. JUNI 1974

```
NOLNZC (12), NFKZZC (6), NBENZC (30),
OLN(3), FKZ(2), BEN(6)
NBKZ1 (6), NBKZ2 (6), NBKZ3 (6),
NBKZ4 (6),
BKZ1 (2), BKZ2 (2), BKZ3 (2), BKZ4 (2),
NRKSP, NRTSP, NRPSP,
MV (2), KENN (2), MVOP (3), DATUM (3)
```

b) in Algol:

```
common KENDAT
integer NKSB, NBGB, NTSB, NPSB, NDRS,
        NSBG, NRZS, NANR, NTYP, NGNR,
        NSNR, NKSPMX, NTSPMX, NPSPMX;
integer array NOLNZC [1:12] , NFKZZC [1:6] , NBENZC [1:30] ,
              OLN [1:3], FKZ [1:2], BEN [1:6],
              NBKZ1, NBKZ2, NBKZ3, NBKZ4 [1:6],
              BKZ1, BKZ2, BKZ3, BKZ4 [1:2];
integer NRKSP, NRTSP, NRPSP;
integer array MV, KENN [1:2],
              MVOP, DATUM [1:3];
```

Dabei bedeuten:

α) NKSB Kernspeicherbedarf des Abschnittskommandos

NBGB Bandgerätebedarf

NTSB Trommelspeicherbedarf

NPSB Plattenspeicherbedarf

NDRS Druckerseitenschranke

NSBG Speicherbedarfsgruppe

NRZS Rechenzeitschranke (in Sekunden!)

NANR Auftragsnummer

NTYP Typ des Eingabegerätes:

- 2 = Sichtgerät
- 0 = Fernschreiber
- 15 = Lochkartenleser
- 14 = Lochstreifenleser

NGNR Gerätenummer

NSNR Stationsnummer

NKSPMX bisher maximal benötigter Kernspeicher

NTSPMX " " " Trommelspeicher

NPSPMX " " " Plattenspeicher

NRKSP	restlicher	zur	Verfügung	stehender	Kernspeicher
NRTSP	"	"	"	"	Trommelspeicher
NRPSP	"	"	"	"	Plattenspeicher

(in K Ganzworten)

Diese Zahlen sind normale Integer-Größen, je nachdem, ob KENDAT von Algol oder von Fortran aus aufgerufen wurde, nach Algol- oder Fortran-Konventionen.

β) NOLNZC Operatorlaufname
NFKZZC FKZ des Abschnitts
NBENZC BEN des Abschnitts
NBKZ1 Das 1. Benutzerkennzeichen des Abschnitts
⋮ ⋮
NBKZ4 Das 4. Benutzerkennzeichen des Abschnitts
Diese Felder sind Integer-Arrays, in die der Operatorlaufname etc. abgelegt wird, und zwar jeweils ein ZC-Zeichen als Integer-Zahl pro Ganzwort (immer entsprechend den Konventionen der Sprache des aufrufenden Programms)

χ) OLN Operatorlaufname
FKZ FKZ des Abschnitts
BEN BEN des Abschnitts
BKZ1 Das 1. Benutzerkennzeichen des Abschnitts
⋮ ⋮
BKZ4 Das 4. Benutzerkennzeichen des Abschnitts
MV Die Maintenance-Version
KENN Ein Rechenzentrum-spezifisches Kennwort
MVOP MV-Nummer des Operators vom Montiere-Kommando
DATUM Tagesdatum in druckfertiger Form
Diese Felder sind echte Strings, und zwar beim Aufruf von Algol aus Algol-strings, beim Aufruf von Fortran aus Fortran-strings (im Sinne von string-handling)

Dabei ist noch folgendes zu beachten:

Die Angaben unter BEN= und FKZ= werden immer mit Leerzeichen aufgefüllt, d. h. NBENZC und NFKZZC sind mit der Zahl "175" aufgefüllt, alle anderen Felder unter β) sind mit "O" aufgefüllt.

4. Arbeitsweise

4.1 Es werden die Kenndaten des SSR 4 O benutzt, die BKZ's werden mit dem SSR 253 32 (IS=4) erfragt, das Datum mit dem SSR 4 32 (T=2), die restlichen zur Verfügung stehenden Speicherberechtigungen mit dem SSR 4 28. Dabei ist noch zu berücksichtigen, daß der Wert von NRKSP in Algol meistens kleiner ist als der Wert, den die Prozedur MEMORY liefert, da der aktuelle Wert des Freispeicherpegels nicht berücksichtigt wird, d. h. NRKSP hat als Wert den tatsächlichen freien Kernspeicher, der z. B. durch ein neues Gebiet belegt werden kann, während MEMORY den Wert liefert, der durch ein neues Algol-array im Freispeicher belegt werden kann.

Bemerkung: Für BCPL-Aufruf gilt für die Ablage der einzelnen Größen dasselbe wie beim Fortran-Aufruf. Das heißt also, die Zahlen werden als Festkommagrößen abgelegt, die Strings als Fortran-strings.
Wird KENDAT als Funktionsprozedur aufgerufen, so ist der Funktionswert die Anfangsadresse des Common-Bereiches (besonders für BCPL-Aufrufe gedacht). Diese Anfangsadresse steht für BCPL-Programme aber ebenso in der Zelle mit dem Namen B.KENNDATEN (Halbwort, während alle anderen Größen auch für BCPL in Ganzworten liegen).

Speziell für BCPL wurde noch eine Anzahl von Kontaktnamen geschaffen. Es gibt folgende Kontaktnamen, deren Bedeutung aus den vorigen Seiten ersichtlich ist:

NKSB, NBGB, NTSB, NPSB, NDRS, NSBG, NRZS, NANR, NTYP,
NGNR, NSNR, NKSPMX, NTSPMS, NPSPMX, NOLNZC, NFKZZC,
NBENZC, OLN, FKZ, BEN, NBKZ1, NBKZ2, NBKZ3, NBKZ4,
BKZ1, BKZ2, BKZ3, BKZ4, NRKSP, NRTSP, NRPSP, BMV, BKENN,
MVOP, BDATUM,

Größen, die Zahlen enthalten, können dabei z. B. so angesprochen werden: NKSB!1 , da die Kontaktnamen die (Ganzwort-) Adresse der zugehörigen Größen enthalten.

Strings sind normal unter dem Kontaktnamen anzugeben.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Benutzungsbeschreibung	B0.E5.02 B
Programm-Thema: Ausführung von vorrangigen PS-Kommandos in einem Operatorlauf	Prog.-Name: KOMMDO

Gliederung:

1. ZWECK DES PROGRAMMS

Das Unterprogramm ermöglicht es, von einem Operator aus Programmiersystem-Kommandos ausführen zu lassen und dann den Operator fortzusetzen.

2. AUFBAU

2.1 Programmiersprache: TAS

2.2 Programmierform: Prozedur

3. HANDHABUNG

3.1 Deklaration:

a) in ALGOL60: procedure KOMMDO(X); code;

b) in FORTRAN: nicht nötig

c) in COBOL: nicht nötig

d) in BCPL:
```
EXTERNAL KOMMDO : B.KOMMDO, B.FLUCHT
NONREC 12:B.KOMMDO
```

e) in TAS:
```
EXTERN KOMMDO,
```

3.2 Aufruf

a) in ALGOL60: KOMMDO(S [,LAB1 [,LAB2]]);

b) in FORTRAN: CALL KOMMDO(S [,LAB1[, LAB2]])

c) in COBOL: ENTER TAS KOMMDO USING S [LAB1 [LAB2]].

d) in BCPL: B.KOMMDO(S,LAB1,LAB2)

e) in TAS:
```
SFB KOMMDO,
```
dabei muß in RA die Anfangsadresse eines Versorgungsblockes stehen, der folgendermaßen aufgebaut sein muß:

2	0	SS^4	PZ^8
2	TADR	LNG	
2	L1	ZV	
2	L2	ZV	

SS = Sprachschlüssel = 8 für TAS
PZ = Parameterzahl ($1 \leq PZ \leq 3$)
TADR = Anfangsadresse des Textes, muß Ganzwortadresse sein.
LNG = Anzahl der Oktaden
L1 = erste Fehleradresse
L2 = zweite Fehleradresse
ZV = Zusatzversorgung (muß = 6 sein, wie in Fortran für Labels)

Ist $PZ < 3$, so kann das vierte Ganzwort fehlen,
ist $PZ = 1$, so kann auch das dritte Ganzwort fehlen.
Der Versorgungsblock kann schreibgeschützt sein.

Bedeutung der Parameter:

LAB1, LAB2 sind Fehlerlabel (siehe 5. Fehlerbehandlung)

S steht für den Text. S kann folgendes sein:

a) in ALGOL60: α) ein String, in Stringquotes eingeschlossen
β) ein Feld, auf das ein Text eingelesen wurde
γ) ein Feldelement, ab dem ein String beginnt
δ) ein Feld, auf das im A-Format eingelesen wurde und das genau 80 Zeichen enthält.

b) in FORTRAN: α) eine Literalkonstante
β) eine Variable, auf die im A4-Format eingelesen wurde
γ) ein eindimensionales INTEGER*4 oder REAL*4 Feld, das eine Literalkonstante enthält.
δ) ein LOGICAL*1 Feld, das je Element ein Zeichen enthält.

c) in COBOL: α) alphabetisches Feld
β) alphanumerisches Literal
γ) alphanumerisches Feld
(Die Felder müssen elementar sein.)

d) in BCPL: BCPL-String; die Stringlänge darf anstatt in der ersten Oktade auch im gesamten ersten Wort stehen.

e) in TAS: eine Oktadenfolge, die auf Ganzwortgrenze beginnt.

3.3 Speicherbedarf:

320	Befehle
9	Ganzworte Konstanten
79	Ganzworte Arbeitsspeicher

4. ARBEITSWEISE

4.1 Verfahren

Es wird ein Startsatz für den Entschlüßler aufgebaut, und dieser mit einem zufälligen Operatorlaufnamen gestartet. Vor den Text wird ein Fluchtsymbol gesetzt und es wird eine Fluchtsymbolverweisliste (FLULI) erstellt. Ist das zweite Fehlerlabel angegeben, so wird zusätzlich nach dem Kommando (im selben Entschlüßlerlauf) noch folgende Kommandos ausgeführt:

```
¤WAHLSCHALTER,WS1,-
¤SPRINGE, ENDE,(FE2)
¤WAHLSCHALTER,-,WS1
```

Diese Kommandos bewirken, daß bei einem aufgetretenen Fehler während der Ausführung des Kommandos (z. B. Operatorlauf mit Fehler beendet) der Wahlschalter WS1 gesetzt und sonst gelöscht wird. Dieser Wahlschalter wird nach Beendigung des Entschlüßlerlaufes abgefragt für die Fehlerbehandlung (siehe 5.) und dann wieder auf seinen ursprünglichen Zustand gebracht.

In BCPL sind variable Parameterzahlen nicht möglich; der Aufruf muß stets mit 3 Parametern erfolgen. Bei LAB1, LAB2 bedeutet binär Null jedoch Undefiniert.

4.2 Gültigkeit:

Die Zeichenzahl LNG des Textes ist beliebig (>0).

Erlaubt sind alle Zentralcodezeichen (Ignores werden entfernt); zusätzlich erlaubt sind Ersatzdarstellungen der Form <Fls>{<Ziffer>}^{3} entsprechend der Syntax der Kommandosprache - mit der Erweiterung, daß auch Steuerzeichen (<64) so dargestellt werden können.

<Fls>[und <Fls>] wirken als Stringklammern: Innerhalb eines so eingeklammerten Strings wird, abgesehen von weiteren, geschachtelten Stringklammern, <Fls> nicht erkannt. Das äußerste Klammerpaar wird entfernt.

5. FEHLERBEHANDLUNG

Es gibt zwei verschiedene Arten von möglichen Fehlern:

1. formale (Versogungs-) Fehler:
   a) falsche Zeichenzahl (LNG=0)
   b) falscher Parametertyp
   c) Entschlüßler nicht startbar, da Operatorlauf-Verschachtelung bereits zu tief oder aus ähnlichen Gründen
   d) falsche Klammerstruktur von <Fls>[und <Fls>]
   e) Startsatz für PS&ENTSCHL wird zu lang (mehr als 250 Fluchtsymbole)

2. Fehler bei Ausführung des Kommandos
   (z. B. Operatorlauf mit Fehler beendet)

   Tritt einer der Fehler unter 1. auf, so wird zunächst ausgedruckt:

   K O M M D O : FEHLERHAFTER AUFRUF .

   Ist dabei das erste Fehlerlabel nicht angegeben oder der 2. Parameter vom falschen Typ, so wird anschließend ausgedruckt:

   K O M M D O : KEIN FEHLERLABEL VORHANDEN ,

andernfalls wird auf das erste Fehlerlabel gesprungen.
Bei einem Fehler 2. Art wird auf das zweite Label gesprungen, falls es angegeben wurde, sonst wird KOMMDO normal beendet.

6. ERGÄNZUNG

Der Text S darf maximal 250 Kommandos enthalten,

Welches Zeichen als Fluchtsymbol erkannt wird, ist umsteuerbar - voreingestellt ist das

"KISSEN"(□ = ZC-Wert 124).

Umgestellt wird dieses Fluchtsymbol durch Belegen einer Common-Variablen mit dem entsprechenden Zentralcodezeichen:

a) in ALGOL60 :

```
common  FLUCHT
integer FLUSY;
        :
FLUSY :=  ⟨Zentralcode-Wert⟩
        :
```

b) in FORTRAN :

```
COMMON/FLUCHT/IFLUSY
      :
IFLUSY = ⟨Zentralcode-Wert⟩
      :
```

c) in COBOL : nicht möglich

d) in BCPL :

```
B.FLUCHT := ⟨Zentralcode-Wert⟩
```

e) in TAS :

```
FLUCHT = CZONE V,
         ABLAGE FLUCHT(VO),
FLUSY  = ASP 2/G,
         AEND(VO),
        :
BA ⟨Zentralcode-Wert⟩,  C FLUSY,
        :
```

Das erste Fluchtsymbol des Textes darf aus Kompatibilitätsgründen zu älteren Versionen von KOMMDO fehlen.

Es existiert außerdem der Eingang KOMMD2, der genauso wie KOMMDO versorgt wird, bei dem der Entschlüßler aber nicht mit SSR O 4, sondern mit SSR O 8 gestartet wird.

Da in diesem Fall die Endebehandlung des Operators umgangen wird und somit Fehler auftreten können, die der "Normalbenutzer" nicht interpretieren kann (kein Leeren der E/A-Puffer etc.), ist dieser Eingang mit Vorsicht zu genießen.

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung B0.E2.13 B

Programm-Thema:
Löschung einer über symb. Gerätenr. identifiz. Datei

Prog.-Name
LOEDAT

Gliederung:

1. Durch Angabe der symbolischen Gerätenummer, die im Starte-Kommando einer Datei zugeordnet wurde (Spezifikation DATEI =...) wird die angegebene Datei gelöscht unter Berücksichtigung einer eventuellen GV-Nummer.

2.1. TAS

2.2. procedure

3.1. in Algol: procedure LOEDAT(X); code;
in Fortran: ALGOL EXTERNAL LOEDAT

3.2. als eigentliche Prozedur

In Algol: LOEDAT(15,MARK);
15 = Gerätenummer (auch Variablen zulässig)
MARK = Marke, auf die gesprungen wird bei einem Fehler, darf auch fehlen - dann wird bei Fehler normal fortgefahren.

3.3. 26 Ganzworte

3.4. -

4.1. mit SSR 253 4 wird die Datei, der die angegebene symbolische Gerätenummer im Starte-Kommando zugeordnet wurde, gelöscht. Dnummer-Angaben werden dabei berücksichtigt. Die Datei muß von der Bearbeitung abgemeldet sein (siehe CLODA / CLOSE)

4.2. entfällt

4.3. entfällt

5. Es werden alle SSR- und sonstigen Fehler abgefangen. Ist ein Fehlerlabel als zweiter Parameter angegeben, so wird im Falle eines Fehlers dieses angesprungen, sonst wird immer normal fort-

Datum: 29. JUNI 1973

1. Zweck des Programms

NDATEI macht eine Datei, die nicht im Starte-Kommando unter Datei=... angegeben ist, für alle Algol- und Fortran-Routinen, die eine Datei mit symbolischer Gerätenummer ansprechen, zugänglich.

2. Aufbau

2.1 Programmiersprache: TAS

2.2 Programmform:
a) in Algol: boolean procedure
b) in Fortran: integer function

3. Handhabung

3.1 Deklaration:

a) in Algol:
```
'boolean' 'procedure' NDATEI (SGNR,
DTN,PASS,DBN); 'code';
```

b) in Fortran:
nicht nötig

3.2 Aufruf:

a) in Algol: In booleschen Ausdrücken, z.B.:
```
'if' 'not' NDATEI (17,'('DATEINAME')',
0,'('DB')') 'then' 'goto' FEHLER;
```

b) in Fortran: Als Funktionsprozedur in arithmetischen Ausdrücken, z.B.:
```
IF (NDATEI(81,S1,S2,S3).EQ.0) GOTO 100
```

Der letzte oder die beiden letzten Parameter können fehlen.

Bedeutung der Parameter:

SGNR symbolische Gerätenummer, unter der die Datei ansprechbar sein soll.

Typ:
a) in Algol: integer-Ausdruck
b) in Fortran: integer*4 Variable oder konstante

DTN Dateiname, mit oder ohne Generations-Versionsnummer. Syntax wie im Kommando, nur werden Leerzeichen nicht überlesen, sondern mit ausgewertet. Der Dateiname darf beliebige Zeichen enthalten, muß also kein Standardname sein.

Typ: a) in Algol: Algol-string z.B. '('MAX')' oder array-Name, auf dem ein String steht.

b) in Fortran: String oder Literal-konstante.

PASS Passwort der Datei: Beliebige Zeichenfolge, maximal die ersten 6 Zeichen werden ausgewertet. Hat die Prozedur beim Aufruf nur 2 Parameter, oder ist Pass=0 oder Leerstring, so wird kein Passwort angenommen. Typ siehe DTN.

DBN Datenbasisname der Datei: beliebige Zeichenfolge, maximal die ersten 6 Zeichen werden ausgewertet. Fehlt der 4. Parameter, so wird &STDDB eingesetzt.
Typ siehe DTN

3.3 Speicherbedarf:

308 Befehle
14 Ganzworte Konstanten
10 Ganzworte Arbeitsspeicher

3.4 Zeitbedarf:

0.5 msec

4. Arbeitsweise

4.1 Verfahren:

Es wird ein neues Dateilistenelement in den Startsatz, der in der Czone S&CL auf S&C1 ff. liegt, eingetragen. War noch kein Listenverweis der Datei-position des Startekommandos vorhanden, so wird eine Liste angelegt.

5. Fehlerbehandlung:

Tritt ein Fehler auf, so wird beim Aufruf in Fortran als Funktionswert eine 0 übergeben, beim Aufruf in Algol der Wert 'FALSE'.
Als Fehler sind möglich:

a) Der Startsatz war bereits zu lang
b) Es war kein Dateiname vorhanden
c) Es war keine symbolische Gerätenummer vorhanden
d) Eine vorhandene Generations-Versions-nummer war fehlerhaft.

Bei richtiger Ausführung wird in Fortran eine 1, in Algol der Wert 'TRUE' als Funktionswert übergeben.

Es existieren zwei weitere Eingänge in

NDATEI: SDATEI und UDATEI .

Die Parameter sind die gleichen wie bei NDATEI, die Eingänge ermöglichen es jedoch, unter derselben symbolischen Gerätenummer nacheinander verschiedene Dateien zu bearbeiten (insbesondere also beliebig viele Dateien in einem Programmlauf).

SDATEI: Mit diesem Aufruf wird, unabhängig davon ob die symbolischen Gerätenummern übereinstimmen, der letzte durch NDATEI oder SDATEI durchgeführte Eintrag überschrieben.

Das verhindert z. B. auch einen Überlauf des Startsatzes.

UDATEI: Mit diesem Aufruf wird das Element im Startsatz mit derselben symbolischen Gerätenummer überschrieben. Ein solches Element muß bereits existieren, d. h.: Ist der Dateiname länger als 6 Zeichen, so muß das Element vorher schon mal durch einen Dateinamen mit mehr als 6 Zeichen belegt gewesen sein; ist eine GV-Nummer vorhanden, so muß das Element vorher schon mal durch eine Datei mit GV-Nummer belegt gewesen sein.

1. Zweck des Programms
Auslieferung des UEBWS-Spezifikationswertes des STARTE-Kommandos

2. Aufbau

2.1 Programmiersprache: TAS
Programmierform: integer procedure bzw. INTEGER FUNCTION

3. Handhabung

3.1 Deklaration

3.1.1 in ALGOL: integer procedure NUEBWS; code;

3.1.2 in FORTRAN: nicht nötig

3.2 Aufruf

3.2.1 in ALGOL: I:= NUEBWS;

3.2.2 in FORTRAN: K = NUEBWS(o);

(Der Parameter ist bedeutungslos, muß aber aus syntaktischen Gründen angegeben werden)

3.3 Speicherbedarf ca. 20 Ganzworte

4.1 Verfahren:
Aus dem Startsatz in S&C1 wird der entsprechende Parameter genommen und für Algol evl. noch gewandelt.
Der Funktionswert ergibt sich wie folgt:

UEBWS(STARTE)	NUEBWS
"undefiniert"	0
n	n
n BTR	-n
BTR	-10000

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung BØ.E3.06 B

Programm-Thema: Auswertung der Spezifikation DNUMMER (STARTE)

Prog.-Name: NUMMD

Gliederung:

1. Zweck des Programms

Die Besetzung der Spezifikation DNUMMER(STARTE) wird ausgewertet.

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmierform: INTEGER FUNCTION

3. Handhabung

3.1. Deklaration: In FORTRAN nicht nötig,
in ALGOL: integer procedure NUMMD(Z); fortran;

3.2. Aufruf mit genau einem Parameter vom Typ INTEGER*4, integer oder INTEGER*2:
m=NUMMD(n)

m erhält den Wert k, wenn eine Angabe DNUMMER = nÜk gemacht war, den Wert -1, falls n durch DNUMMER = kÜn schon vergeben ist, den Wert n sonst.

4.2. Gültigkeitsbereich

Der Parameter muß zwischen 1 und 99 einschließlich liegen.

5. Fehlerbehandlung

Ist der Parameter von falschem Typ oder negativ oder größer als 99, so ist der Funktionswert Null.

Datum: 27. JUNI 1973

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung BØ E1 . 06 B

Programm-Thema: Auswertung des Operatorlaufnamens in FORTRAN

Prog.-Name: ØLNAME

Gliederung:

1. Zweck des Programms

OLNAME soll den Operatorlaufnamen (Spez.*LAUF (STAPTE)) FORTRAN-Programmen zugänglich machen.

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmierform: SUBRØUTINE

3. Handhabung

3.1. Deklaration des Programmes ist nicht nötig; der Parameter muß ein Feld von mindestens 2 GW Länge haben (z. B. DIMENSION S(2)).

3.2. Aufruf: CALL OLNAME (S)
Nach dem Aufruf steht der Laufname auf dem Feld S als String im Sinne von Stringhandling (also nicht als Literal) zur Verfügung.

Datum: 27. JUNI 1973

Programm-Thema: Schreiben eines beliebigen Startsatzes in eine Datei

Prog.-Name: RB&BASTEL

Gliederung:

1. Der Operator schreibt seinen Startsatz so in eine Datei, daß er für Algol- und Fortran-Programme weiterverarbeitbar ist und startet danach evtl. einen angebbaren Operator mit Standard-Startsatz.

2.1. TAS

2.2. Operator

3.1. entfällt

3.2. Start durch definiertes Kommando

3.3. 2K Ganzworte

3.4. minimal

4.1. Der Operator kreiert zunächst eine Datei: BASTEL&DATEI, RAN, U1, U8ØØ, P

In diese Datei schreibt er sodann seinen gesamten Startsatz in folgender Weise:

1. Satz: Anzahl der Spezifikationen (3-stellig, also im I3-Format lesbar)

2. Satz: Anzahl der Teilwerte der 1. Spezifikation (3-stellig) mögliche Werte:

-1 Spezifikation = -STD-
0 Spezifikation = "undefiniert"
N > o Anzahl der Teilwerte

3. Satz bis (N+2).Satz: Parametertyp (3-stellig), dahinter die Zeichenzahl des Parameters (3-stellig), dahinter alle Zeichen des Parameters, jedoch maximal 999 Zeichen. Ist der Parame-

ter ein Fremdstring, so werden Zeilenwechsel als Oktade 255 abgelegt.

(N+3.)Satz: Anzahl der Teilwerte der 2. Spezifikation (3-stellig)
.
.
.
etc.

An Spezifikationstypen kann auftreten:

Typ = 3, 4, 5, 6, 7 und als Spezialtyp 2

Typ = 3 tritt auf, wenn die Spezifikation mit NZ4 definiert ist. Die Zahl wird dann immer als 5 Zeichen abgelegt, ist also z. B. direkt mit I5-Format lesbar.

Typ = 7 Bei Fremdstring im Gebiet, der meistens nur im Abschnitt vorkommt, wird zusätzlich die erste Zeile, wenn sie nur aus Leerzeichen besteht, entfernt.

War das Kommando mit einem Eingang definiert, so wird nach Erstellen der BASTEL&DATEI ein Standardstartsatz aufgebaut und mit diesem derjenige Operator gestartet, dessen Name auf der letzten Spezifikation als Spezifikationswert angegeben ist, (die letzte Spezifikation muß mit (NL, SN) definiert werden!). Dieser Startsatz ist der gleiche wie einer, der durch folgendes Starte-Kommando erzeugt würde:

```
□STARTE, OP, DUMP   =  T-ALLES'F-NEST'A-NEST'C-TEIL,
             UEBWS  =  4095,
             DATEI  =  99-BASTEL&DATEI (g.v),
             AKTIV  =  ALLE
```

(g.v) ist die Generations-Versionsnummer der kreierten Datei BASTEL&DATEI. Tritt einmal oder mehrmals der Typ 7 (Gebietsfremdstring) auf, so ist der zuletzt angegebene

mit Gerätenummer 5 lesbar, als wenn er beim Starte-Kommando unter DATEN=/... angegeben wäre. War der Eingang > 50, so wird zusätzlich der Operatorlaufname verändert, wenn das Kommando rekursiv gegeben wurde. Bei einem Eingang > 50 wird die Spezifikation DUMP außerdem "undefiniert" gelassen. Als Spezifikations- (Teil-) Werte sind bei der Kommandodefinition folgende Typen erlaubt:

NL
F
STD
N
QN
SN
NZ4
DT
NDT

Der Benutzer muß selber dafür sorgen, daß die jedesmal neu kreierten Dateien BASTEL&DATEI (g.v) wieder gelöscht werden, z. B. durch den Unterprogrammaufruf LOEDAT(99); (siehe BO.E2.13).

Außerdem steht der komplette Startsatz in unveränderter Form in den ersten beiden Achtelseiten des (evtl. kreierten) Gebietes mit dem prozeßspezifischen Gebietsnamen RB&BAS (1 K Länge, Träger= PLATTE), aus dem sich der TAS-Programmierer noch Informationen holen kann.

Es gelten folgende Zuordnungen von Spezifikationswerten zu Typen, unabhängig davon, ob die jeweiligen Spezifikationsdefinitionen erlaubt sind:

TYP	SPEZIFIKATIONSWERT IM DEFINIERE-Kommando
2	DT, NDT
3	NZ4, TAB(I)
4	INU
5	SN, QN, FZ, NZ, PZ, TR
6	N, F,
7	F

Dazu ist folgendes zu beachten:

1. Wird bei der Kommandodefinition DT oder NDT angegeben, so wird die Zeichenfolge des Dateinamens in der BASTEL&DATEI abgelegt.

Eine eventuell vorhandene Nummer davor wird entfernt, ebenfalls der Datenbasisname, wenn er ■ &STDDB ist.

Zum Beispiel: 1-&STDDB.MAX(1.3) wird als MAX(0001.03) abgelegt.

Ein angegebenes Passwort wird ebenfalls weitergegeben. Der spezielle TYP 2 wurde deshalb hinzugefügt, um bei einer Spezifikation (F,DT) leichter zwischen Dateiangabe und Fremdstring unterscheiden zu können.

2. Wird eine Spezifikationsdefinition genommen, die nicht ausdrücklich als erlaubt gekennzeichnet ist (s. o.), so werden die Werte binär in der BASTEL&DATEI abgelegt, und der Programmierer muß sie selbst wandeln.

Für den ALGOL60-Programmierer stehen weitere Hilfsprozeduren zur Auswertung einer BASTEL&DATEI zur Verfügung (in Bibliothek BOGOL):

1. integer procedure STARTSATZ(X); code;

   Es gibt 4 verschiedene Aufrufarten:

   a) I:=STARTSATZ(-1 , Fehlerlabel);
      liefert als Funktionswert die Eingangsgröße

   b) I:=STARTSATZ(0, Fehlerlabel);
      liefert als Funktionswert die Anzahl der Spezifikationen des definierten Kommandos (ohne den letzten - den Operatornamen - und ohne den Eingang).

   c) I:=STARTSATZ(N,Fehlerlabel);
      liefert als Funktionswert die Anzahl der Teilwerte der N-ten Spezifikation, d. h.:

      bei "undefiniert" =0 und
      bei -STD- =-1

   d) TYP:=STARTSATZ(N,I,FELD,Fehlerlabel);
      Funktionswert ist der Typ des I-ten Teilwertes der N-ten Spezifikation des Kommandos. Die Zeichenfolge dieses Teilwertes wird in dem als integer array FELD[1:170]; zu deklarierenden Feld auf dritter Parameterposition als String abgelegt.

   Intern wird aus der BASTEL&DATEI mit BODAT-Prozeduren (BO.CO.09) gelesen. Tritt irgendein Fehler auf, so wird auf das Fehlerlabel gesprungen. Fehlt dieses, so wird das Programm mit Fehlermeldung abgebrochen, wenn ein Fehler auftritt.

2. procedure SSBEREICH(X); code;

Dient zur Auswertung von Bereichsangaben aus der BASTEL&DATEI, die mit (N) definiert sind:

SSBEREICH(N, I, ZAHL1, ZAHL2, Fehlerlabel);

Der Aufruf bewirkt, daß der I-te Teilwert der N-ten Spezifikation als Bereichsangabe ausgewertet wird, wobei ZAHL1 die Zahl vor dem Strich und ZAHL2 die Zahl nach dem Strich zugewiesen wird. Fehlt eine der Zahlen, so wird dem entsprechenden Parameter eine Null zugewiesen. Insbesondere wird, wenn als Teilwert nur eine Zahl angegeben ist, diese Zahl der Variablen ZAHL1 zugewiesen und ZAHL2 erhält als Wert 0. Das Fehlerlabel muß angegeben werden

3. boolean procedure SSDATEI(N, I, SGNR, MODUS);

bewirkt, daß der I-te Teilwert der N-ten Spezifikation als Dateiname ausgewertet (Definition: DT) und unter der symbolischen Gerätenummer SGNR bekannt gemacht wird. Ist MODUS≠0, so wird zusätzlich die letzte auf diese Weise bekanntgemachte Datei wieder aus dem Startsatz entfernt, so daß der Startsatz nicht überlaufen kann.

Der Funktionswert ist true, wenn kein Fehler auftrat, bei irgendeinem Fehler false.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Benutzungsbeschreibung	BØ.E2.17 B
Programm-Thema: Reservieren einer Datei über symbolische Gerätenummer	Prog.-Name: RESERV

Gliederung:

1. Zweck des Programms

Die über eine symbolische Gerätenummer identifizierte Datei wird reserviert wie im RESERVIERE-Kommando.

2. Aufbau

2.1. Programmiersprache: TAS

2.2. Programmierform: Prozedur oder Funktionsprozedur

3. Handhabung

3.1. Deklaration: `procedure RESERV(X); code;`

oder: `integer procedure RESERV(X); code;`

3.2. Aufruf:

`[I:=] RESERV ( SGNR [,RESERVE [,LABEL]]);`

Bedeutung der Parameter:

SGNR = Symbolische Gerätenummer der Datei im Startekommando

RESERVE = Anzahl der Sätze, für die noch Platz sein soll in der Datei. Fehlt der Parameter, so wird er zu 0 ergänzt.

LABEL = Label oder Marke, auf die im Fehlerfall gespungen werden soll.

Funktionswert = Anzahl der von der Datei nach der Reservierung belegten K's.

Tritt ein Fehler auf und ist kein Fehlerlabel angegeben, so ist der Funktionswert = 0.

Datum: 07. Aug. 1974

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Konstruktionsbeschreibung	BØ.E3.15 K
Programm-Thema: "Stumme" Version von S&DPRO	Prog.-Name S&DPRO

Die in der Bibliothek UNIHIP liegende Version unterscheidet sich von der &OEFDB-Version des Unterprogramms S&DPRO nur dadurch, daß der sonst unvermeidliche Text

GEAENDERT: DB.DATEI(1.0)

unterdrückt wird.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Konstruktionsbeschreibung	BØ.Co.07 K
Programm-Thema: "Gib Zeile von Konsole!"	Prog.-Name S&GZK

In der Bibliothek UNIHIP befindet sich eine Abwandlung des System-Unterprogramms S&GZK:

1. Kernspeicherbedarf
   Die geänderte Version benötigt etwa 250 G.. KSP weniger Variablen, zusätzlich ist der verbleibende Speicherbedarf möglichst in den 22-bit Adressenraum gelegt worden. Für die Indexzellen wurde eine Indexzone (S&GIBAS) vereinbart. Etwa zehn Befehle sind neu hinzugekommen.
2. Leistung
   a) S&GZK (4.00) erkennt das Eingabeende auf Konsole. Dies führt beim Auftreten des ⊡. auf Konsole dazu, daß sich die EA-Prozeduren in den höheren Programmiersprachen so verhalten wie beim Datei-Ende. Ein anschließender aufruf von S&GZK (4.00) führt zu einer neuen Eingabeanforderung auf Konsole.

   b) Bei gesetztem Zustandswahlschalter 7 ist jetzt auch Lesen von Konsoleingabe mit Fluchtsymbolen möglich. S&GZK (4.00) behandelt die Eingabe wie eine normale Eingabe, die enthaltenen Fluchtsymbole werden als Oktade FL (Dezimalwert 53, hexadezimal '35') übergeben. Der in S&GZK(3.04) auftretende Makro-Alarm unterbleibt.

   c) Enthielt eine Konsoleingabe Kommandos, die durch einen Entschlüsslerlauf ausgeführt wurden, so fragt S&GZK (4.00) nach einer neuen Eingabe mit OPERATORNAME/OPERATORLAUFNAME⊡:. Dabei wird der OLN weggelassen, falls ON = OLN gilt.

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Konstruktionsbeschreibung	BØ.E3.14 K
Programm-Thema: "Stumme" Version von S&OPZEIT	Prog.-Name: S&OPZEIT

Die in der Bibliothek UNIHIP liegende Version von S&OPZEIT unterscheidet sich von der Version in der &OEFDB nur dadurch, daß die Eingänge rel. 0 und rel. 1 kurzgeschlossen sind. Wird diese Version an ein in einer höheren Sprache geschriebenes Programm anmontiert, unterbleiben also die sonst unbedingt erfolgenden Meldungen

START Programmname (1.0)

und

ENDE Programmname (1.0) 2.32 .

Die anderen Eingänge (die auch von BEENDE (BO.E3.31) benutzt werden) funktionieren normal.

SENDE

Spezifikation:

1	QUELLE	Angaben der gewünschten Datei oder eines Fremdstrings
2	BEREICH	Angabe der zu bearbeitenden Sätze
3	TRAEGER	Bezeichnung des Datenträgers der Datei
4	MAL	Auszeichnung eines Zeichens als Mal (Fluchtsymbol-Ersatzzeichen)
5	PRØTØKØLL	Steuerung der Protokollierung
6	KENNZEICHEN	Kennzeichnung der Sendung
7	VERFALLSZEIT	Verfallszeit der Sendung

Kommando für das Programmiersystem |

Bemerkung: Die ersten 5 Spezifikationen sind identisch mit denen des Kommandos TU, da sie auch von demselben Operator, BO&TUE ausgewertet werden.

Außer Dateien können auch Fremdstrings bearbeitet werden.
Ersatzdarstellungen der Form <MAL><Ziffer><Ziffer> Ziffer mit Werten ≤ 6, Stringklammern der Form <MAL>[und <MAL>] sowie <MAL>(und <MAL>) werden ausgewertet.

Die zusätzliche Spezifikation Träger erlaubt Bearbeitung nicht eingeschleuster Dateien.

Als <MAL> ist ein beliebiges Zeichen zulässig.

Es können mehrere, nicht notwendige disjunkte, Bereiche angegeben werden, die in der angegebenen Reihenfolge (evtl. mehrmals) ausgeführt werden.

Wirkung: Die Sätze der angegebenen Bereiche in der bezeichneten Datei auf dem angegebenen Träger werden gemäß der Spezifikation MAL auf Fluchtsymbole untersucht und nach Auswertung etwaiger Ersatzdarstellungen und Stringklammern in ein Hintergrundgebiet transportiert, das anschließend als Sendung abgeschickt wird (einschl. Fluchtsymbollisten -Verweis)

Beim Empfänger der Sendung (siehe KDO EMPFANGE) werden diese Kommandos dann ausgeführt durch Starten des Entschlüßlers.
Ist MAL=-STD- angegeben, so werden die Zeilen rein als Text verschickt und beim Empfangen ausgedruckt.

format:

Beispiel

LI SENDE, HOPPLA, 30-118, MAL=115, KENN.= ((*)), VERF.= 100

Der Zeilenbereich 30 - 118 der Datei HOPPLA soll bearbeitet werden. Als MAL wird das Zentralcodezeichen 115 (Oktade '73', Externdarstellung) erkannt.

Als Kennzeichen wird die Zeichenfolge ((*)) genommen, die Verfallszeit soll eine Stunde sein.

QUELLE

/f /	: Die Quelle liegt im Fremdstring f.
datei -p	: Dateiname für die Standard-Datenbasis als Arbeitskatalog. Beim Träger LFD wird das erste auftragsspezifische Benutzerkennzeichen als Verwaltungskatalogname genommen; bei W14 das Standard-Dateimengenkennzeichen.
KAT.datei -p	: Dateiname für den Katalog Kat. Der Katalogname kann ein Datenbasisname, ein Benutzerkennzeichen oder ein Dateimengenkennzeichen sein. p: Eine Externe Datei ist gegen Fremdzugriffe mit dem Paßwort p geschützt.
-STD-	: Es wird nur ein Sendungskopf verschickt, auf den zur Koordination von Aufträgen mittels EMPFANGE gewartet werden kann

obligate Spezifikation zum Kommando SENDE

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Die Datei muß eine Ø-Datei sein.

Wirkung:

Die bezeichnete Datei wird bearbeitet. Liegt die Datei auf externen Speichern (LFD, MB oder WSP), so wird sie zunächst eingeschleust oder angemeldet und vor Ausführung der in ihr enthaltenen Kommandos wieder abgemeldet. Es können auch gesicherte RAM- oder RAN-Dateien direkt von Bändern ohne Verlagerung bearbeitet werden.

SENDE QUELLE

format:

<Wertzuweisung QUELLE> ::= [QUELLE=] {<Fremdstring> | <Katalogname>.<Dateiname>-<Paßwort>}

<Katalogname> ::= <Dateibasisname> | <bkz>

<Dateiname> ::= <Name von Standardlänge> [(<Generationsnr.>.<Versionsnr.>)]

<Generationsnr.> ::= <nat. Zahl n, 1 ≤ n ≤ 9 999>

<Versionsnr> ::= <nat. Zahl n, 0 ≤ n ≤ 99>

<Paßwort> ::= <Normalstring> von 1 bis 6 Zeichen Länge

Beispiel

```
..., QUELLE=LFDBkz. DATEI 17(8.4)-PASS, TRAEGER=LFD
```

SENDE
BEREICH

BEREICH Angabe der Bereiche,die ausgeführt werden sollen

Spezifikation:

Spez.-Wert: "undefiniert" : Die gesamte Datei wird bearbeitet.

n : Der Satz n der angegebenen Datei wird bearbeitet.

a - b : Die Sätze im Bereich a - b werden bearbeitet.

Mehrere Angaben sind durch Apostroph zu trennen

optimale Spez. zum Komm. SENDE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Ist die QUELLE ein Fremdstring oder QUELLE=-STD-, so wird BEREICH nicht ausgewertet.

Wirkung:

Die hier angegebenen Sätze der bezeichneten Datei werden bearbeitet; bei gesicherten RAM-Dateien wird die ursprüngliche Satznumerierung verwendet.
Die Sätze werden genau in der hier angegebenen Reihenfolge bearbeitet - bei überlappenden Bereichen mehrfach.

Fehlt eine Angabe zu BEREICH, so wird die gesamte Datei bearbeitet.

SENDE BEREICH

formal: <Wertzuweisung BEREICH>::= [BEREICH=] <teilwert> [<teilwert>]$^{\infty}$

<Teilwert>::=<nat. Zahl ≠ 0> [-<nat. Zahl≠ 0>]

Beispiel:

... BEREICH = 390, ...

Die Zeile 390 wird als Kommando(folge) interpretiert.

..., BEREICH=13'8-27'3'3

Die angegebenen Sätze werden in dieser Reihenfolge bearbeitet (also die Sätze 13 und 3 je zweimal).

TRAEGER — Angabe eines externen Datenträgers

Spez.-Wert:

MB	(EXDKZ)	Magnetband EXDKZ
W14	(EXDKZ)	Wechselplattenstapel EXDKZ
LFD		Langfristige Datenhaltung

optimale Spezifikation zum Kommando SENDE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Ist die Quelle ein Fremdstring oder Quelle=-STD-, so wird Träger nicht ausgewertet.

Wirkung:

Die Datei wird auf dem entsprechenden Medium gesucht, eingeschleust, gelesen und ausgeschleust.

SENDE TRAEGER

formal:

<Wertzuweisung TRAEGER>::= [TRAEGER=] { MB (exdkz) | W14 (exdkz) | LFD | - }

Beispiel:

..., TRAEGER=MB(900000),...

Die Datei wird direkt vom Magnetband 900000 bearbeitet.

MAL — Definition eines Zeichens als Mal

Spez.-Wert: "undefiniert" : das Zeichen FL (Zentralcodewert 53)

z : das Zeichen z

/z : das (im Normalstring nicht angebbare) Zeichen z

n : das Zeichen mit dem Zentralcodewert n, $16 \leq n \leq 255$,

wird in der Quelle als Mal erkannt

-STD- : Die Quelle enthält keine Kommandos, sondern Text, der beim Empfänger ausgedruckt werden soll

Optionale Spezifikation z. Kdo. SENDE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:
Beim Einlesen der Quelle wird jedes gefundene Mal in Abhängigkeit des unmittelbar darauf folgenden Zeichens interpretiert:

<MAL>[und <MAL>] wirken als Stringklammern: Innerhalb eines so eingeklammerten Strings werden, abgesehen von weiteren, geschachtelten Stringklammern, keine Male erkannt. Das äußerste Klammernpaar wird entfernt.

<MAL>(wird zusammen mit allen folgenden Zeichen bis zum nächsten Zeilenende einschließlich überlesen.

<MAL>) wird zusammen mit allen folgenden Zeichen bis zum nächsten Zeilenwechsel ausschließlich überlesen.

Zeichenfolgen der Form <MAL><Ziffer><Ziffer><Ziffer> werden durch das Zeichen mit dem durch <Ziffer><Ziffer><Ziffer> angegebenen Zentralcodewert z, $z \leq 255$, ersetzt.

In allen anderen Fällen wird <MAL> als Fluchtsymbol in der auszuführenden Kommandofolge interpretiert.

Ist MAL=-STD- angegeben, so wird kein Zeichen als Fluchtsymbol erkannt, sondern die Quelle wird (entsprechend BEREICH etc.) als Textfolge verschickt. Paßt der Text in eine Seite, so wird er als Kernspeichersendung gesendet und es wird nicht auf Rücksendung gewartet, andernfalls wird eine Gebietssendung erzeugt (bei Kdo-Folgen immer Gebietssendung).

SENDE MAL

formal:

<Wertzuweisung MAL> ::= [MAL =]<Zeichen>

<Zeichen> ::= { <Zeichen ≠ {*, =, /, ,, (,), '>}
/ bel. Zeichen [□/]
<natürliche Zahl n, 16≤n≤255> }

Beispiel:

..., MAL=/)

PROTOKOLL

Spez.-Wert:		
	"undefiniert"	keine Protokollierung
	-STD-	Standardprotokollierung gewünscht
	KØ	Kommandoprotokollierung auch auf der Konsole
	A	Anfrage nach weiteren Vorrangkommandos nach Kommandoausführung

optionale Spez. zum Komm. SENDE	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Wird keine Protokollierung gewünscht, so unterbleibt die Protokollierung der Kommandos durch den Entschlüßler. Bei Standardprotokoll werden die Kommandos im Ablaufprotokoll protokolliert, falls es eingeschaltet ist - bei Angabe von KØ zusätzlich auf der Konsole.

Mehrere Angaben sind durch Apostroph zu trennen - "undefiniert" und -STD- sind als Teilwert nicht zugelassen.

Bei MAL=-STD- oder Quelle=-STD- wird diese Spezifikation nicht ausgewertet.

Die Protokollierungsangaben beziehen sich auf das Protokoll des Empfängers des Textes bzw. der Kommandofolge.

SENDE PROTOKOLL

format:

<Wertzuweisung PRØTØKØLL> ::= [PRØTØKØLL=] { -STD- / A / KØ }

Beispiel

..., PRØT.=A

KENNZEICHEN — Kennzeichnung der Sendung

Spez.-Wert:

Normalstring — Zeichenfolge, durch die die Sendung identifiziert wird.

obligate Spezifikation zum Kommando SENDE

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Es werden maximal die ersten 12 Zeichen ausgewertet

Wirkung:

Durch die angegebene Zeichenfolge wird die Sendung mit einem Namen versehen, durch den sie beim Empfangen (siehe Kommando EMPFANGE) eindeutig bezeichnet werden kann, wenn keine andere Sendung mit dem gleichen Kennzeichen vorhanden ist.
Das Kennzeichen sollte deshalb so gewählt werden, daß die Wahrscheinlichkeit gering ist, daß jemand anders das selbe genommen hat.
(KENNZEICHEN wird beim SSR 5 8 als ABS-NAME eingesetzt).

SENDE KENNZEICHEN

formal:

<Wertzuwsg. KENNZEICHEN>::=[KENNZEICHEN =]<Normalstring>

<Normalstring> ::= siehe Kommandohandbuch

Beispiel:

..., KENNZEICHEN = (KE1*KE2),...

VERFALLSZEIT

Spez.-Wert:

"undefiniert"	Die Sendung existiert so lange wie die Warteschlange (ca. 10 Tage)
tt hh mm	tt = Anzahl der Tage hh = Anzahl der Stunden mm = Anzahl der Minuten

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

wenn VERFALLSZEIT ≠ "undefiniert" ist, wird mindestens 1 Minute eingesetzt.

Wirkung:

Die VERFALLSZEIT gibt an, wie lange die Sendung maximal existieren soll.
Außerdem wartet der absendende Auftrag maximal so lange, wie VERFALLSZEIT angibt, auf die Rücksendung des Empfängers, um die Speicherberechtigung zurückzuerhalten. Trifft nach VERFALLSZEIT keine Rücksendung ein, so wird der Fehler

"+++++ Sendung wurde nicht übernommen"

gemeldet und die Sendung wieder gelöscht.
Ist MAL=-STD- angegeben, soll also nur eine zu protokollierende Textfolge gesendet werden und paßt diese Textfolge in eine Seite, so wird eine Kernspeichersendung erzeugt und nicht auf deren Rücksendung gewartet, sondern sofort der Auftrag fortgesetzt.

SENDE VERFALLSZEIT

formal:

$$\langle \text{Wertzuwsg. VERFALLSZEIT} \rangle ::= \left\{ \begin{array}{l} \left[\left[\langle T \rangle \right] \langle H \rangle \right] \langle M \rangle \\ - \end{array} \right\}$$

$\langle T \rangle ::= [t]\ t$
$\langle H \rangle ::= [h]\ h$
$\langle M \rangle ::= [m]\ m$

Beispiel:

```
..., VERF.=    10
 10 Minuten Verfallszeit

..., V.   =   103
  1 Stunde und 3 Minuten Verfallszeit

..., V.   = 30520
  3 Tage, 5 Stunden und 20 Minuten Verfallszeit
```

Benutzungsbeschreibung B0.E1.01 B

Programm-Thema: ABFRAGE VON SIGNALEN

Prog.-Name: SIGNAL

Gliederung:

1. Zweck:

Abfrage eines Signals (vgl. Operateurkommandos SIGS, SIGL)

2. Programmiersprache: TAS, Programmierform: LOGICAL FUNCTION für Fortran

3. Handhabung

3.1 Deklaration: LOGICAL*4 SIGNAL

3.2 Aufruf:
in logischen Ausdrücken ...SIGNAL(n)...
wobei n ($1 \leq n \leq 24$) die Nummer des Signals angibt.
n muß vom Typ INTEGER*4 sein. Der Funktionswert ist .TRUE., wenn das Signal gesetzt ist.

4.1 Es wird der SSR 4 36 benutzt.

4.2 Ist der Parameter $n<1$ oder $n>24$, so ist das Ergebnis stets .FALSE.

Datum: 29. JUNI 1973

PROGRAMMBIBLIOTHEK	RZ UNI BOCHUM
Benutzungsbeschreibung	BØ E5 . 05 B
Programm-Thema: Entschlüßlerstart von ALGOL aus	Prog.-Aut.: TUE

Gliederung:

1. ZWECK DES PROGRAMMS
2. AUFBAU
   - 2.1 Programmiersprache
   - 2.2 programmierform
3. HANDHABUNG
   - 3.1 Deklaration
   - 3.2 Aufruf
   - 3.3 Speicherbedarf
   - 3.4 Zeitbedarf
4. ARBEITSWEISE
   - 4.1 Verfahren
   - 4.2 Gültigkeitsbereich
   - 4.3 Genauigkeit
5. FEHLERBEHANDLUNG

1. Ausführung von Kommandos, die in einer Datei stehen.

2.1: TAS

2.2: procedure

3.1: a) in ALGOL: procedure TUE(x); code;
b) in FORTRAN: ALGOL EXTERNAL TUE

3.2: TUE (SGNR, SATZ1, SATZ2, LABEL1, LABEL2);

Parameter:

SGNR	symbolische Gerätenummer der Datei (siehe Starte-Kommando, Spezifikation, DATEI=...) (DNUMMER-Angaben werden berücksichtigt).
SATZ1	erster Satz der Datei, der getan werden soll. Ist SATZ1=0 oder fehlt der Parameter, so werden alle Sätze der Datei "getan".
SATZ2	letzter zu "tuender" Satz der Datei. Ist SATZ2=0 oder fehlt der Parameter, so werden nur die Kommandos im Satz SATZ1 ausgeführt.
LABEL1	Fehlerlabel (siehe 5.)
LABEL2	Fehlerlabel

Obligat ist nur der erste Parameter (SGNR). Fehlt ein Parameter, so müssen alle weiteren auch fehlen.

4.1: Die Codierung des Mals wird im 1. Wort der Commonzone FLUCHT erwartet
Es wird ein Startsatz für BO&TUE aufgebaut und dieser Operator dann gestartet (siehe Kommando RTU ,...)
Die angegebene Datei muß eingeschleust sein, falls sie LF- oder WSP-Datei ist. (siehe auch Prozedur BO&LOD).

5.: Zwei Fehler werden unterschieden: 1.) BO&TUE ist nicht startbar (Operator nicht vorhanden oder Verschachtelung zu tief)

2.) BO&TUE beendet sich mit Fehler (z.B. Datei nicht vorhanden)

Beim ersten Fehler wird auf LABEL1 gesprungen, sonst auf LABEL2. Ist das entsprechende Fehlerlabel nicht als Parameter vorhanden, so wird das Unterprogramm TUE normal beendet.

Datum: Juli 1978

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung B0.E3.48 B

Programm-Thema: Druckerprotokoll-Manipulationen von Operatoren per Unterprogramm

Prog.-Name: UP&PROTOKOLL

Gliederung:

1. ZWECK DES PROGRAMMS

a) DPSEIT: Ausliefern der Seitenzahl des aktuellen Druckerprotokolls

b) DPAUSG: Druckerprotokoll als Teilauftrag ausgeben und anschließend löschen - zurückgemeldet wird die Teilauftragsnummer.

c) DPKOP: Wie DPAUSG, ohne Löschen des Druckerprotokolls

d) DPABM: Vollständig "Abmelden" des Druckerprotokolls - das heißt: nichts wird mehr ins Ablaufprotokoll gedruckt (auch kein SSR 6 12 oder SSR 6 0) -

e) DPANM: "Anmelden" des Druckerprotokolls, DPABM rückgängig machen.

2. AUFBAU

2.1 Programmiersprache: TAS

2.2 Programmierform: Parameterlose Funktionsprozedur

3. HANDHABUNG

3.1 Deklaration: a) in ALGOL60: integer procedure ⟨name⟩;code;
b) in FORTRAN: INTEGER ⟨Name⟩

⟨Name⟩::= DPSEIT/DPAUSG/DPKOP/DPABM/DPANM

3.2 Aufruf: a) in ALGOL60: I:= ⟨Name⟩;
b) in FORTRAN: I = ⟨Name⟩ (0)
(aus syntaktischen Gründen muß ein Parameter angegeben werden)

Datum: 6. FEB. 1974 .....

3.3 Speicherbedarf:

55 Befehle
14 Ganzworte Arbeitsspeicher
8 Ganzworte Konstanten

3.4 Zeitbedarf: ca. 0,02 sec.

4. ARBEITSWEISE

4.1 Verfahren:

Es wird ein Spezialstartsatz für BO&PROTOKOLL aufgebaut und dieser Operator gestartet. Das Funktionsergebnis liefert BO&PROTOKOLL als Fehlerschlüssel im SSR O 16-Fehlerausgang ab.

Die verschiedenen Eingangsnamen werden folgendermaßen auf die Modusangaben für BO&PROTOKOLL abgebildet:

Name	Modus
DPSEIT	SEITE
DPAUSG	"undefiniert" bzw. -STD-
DPKOP	KOP
DPABM	ABMELD
DPANM	ANMELD

PROGRAMMBIBLIOTHEK RZ UNI BOCHUM

Benutzungsbeschreibung BØ. . B

Programm-Thema: Sofortige Ausgabe auf Konsole ohne Eingabeaufforderung

Prog.-Name: VERDRG

Gliederung:

1. Zweck des Programms

Im allgemeinen erfolgt eine Ausgabe auf Konsole nur in Verbindung mit einer darauf folgenden Eingabeanfrage. Bei längeren Programmläufen kann jedoch auch Anstoß einer Ausgabe sofort bei deren Entstehung (durch WRITE o.ä.) erwünscht sein. Dies wird durch Aufruf von VERDRG erreicht.

2.1. Programmiersprache : TAS

2.2. Programmierform : Montageobjekt für ALGOL und FORTRAN

3.1. Deklaration

a) ALGOL : procedure VERDRG; code;

b) FORTRAN: nicht nötig

3.2. Aufruf

a) ALGOL : VERDRG

b) FORTRAN : CALL VERDRG

3.3. Speicherbedarf,

4 Befehle, 3 GW Konstanten, 1GW Variable

4.1. Verfahren

SSR 6 16, Modus 11 (TKA) mit leerem Auftragspuffer.

Datum: 7.8.1974

ZUSTAND

Abbilden von Zuständen und Situationen auf Wahlschalter

Spezifikation:

1	WAHLSCHALTER	Angabe der zu verändernden Wahlschalter
2	FRAGE	Angabe des zu untersuchenden Zustandes
3	BEDARF	ggf. zu untersuchende Bedarfswerte
4	PRUEFEN	ggf. zu untersuchende Zeichenfolge
5	VERGLEICH	ggf. Vergleichs-strings für PRUEFEN
6	MELDUNG	Angabe zur Protokollierung

Kommando für das Programmiersystem	

Einschränkung

Wirkung:

Der Operator ermöglicht eine variable Steuerung und Verzweigung in Abschnitten und Kommandoprozeduren, indem er prüft, ob eine bestimmte Situation vorliegt, und die angegebenen Wahlschalter, falls ja, setzt und falls nein, löscht. Mit der 6. Spezifikation MELDUNG kann angegeben werden, ob zusätzlich ein JA oder NEIN ausgedruckt werden soll, je nach Situation. Unzulässige Spezifikationswerte bei einer Spezifikation werden grundsätzlich ignoriert, es gibt also keine Fehlermeldung!

ZUSTAND

formal:

⟨ZUSTAND-Kommando⟩ ::= ¤ZUSTAND , [⟨Spezifikationsname⟩ =] ⟨Spezifikationswert⟩

⟨Spezifikationsname⟩ ::= WAHLSCHALTER/FRAGE/BEDARF/PRUEFEN/VERGLEICH/MELDUNG

Beispiel:

```
ZUSTAND, WAHLSCHALTER = WS3'WS7,
         FRAGE = BV5'BV1'PRUEF'STD'BGB'EINGAB,
         BEDARF = 4,
         PRUEFEN = *PARAMETER,
         VERGLEICH = MAX'MORITZ'*87,
         MELDUNG = -STD-
```

Da Teilwertlisten immer von rechts nach links abgearbeitet werden, wird zunächst auf der Konsole angefragt:

FRAGE VON ZUSTAND :
Wird mit JA geantwortet, so werden die Wahlschalter WS3 und WS7 gesetzt. Weitere Prüffolge: Die Wahlschalter 3 und 7 werden gesetzt, wenn:

```
          BGB = 4,

oder      *PARAMETER  =  -STD-
oder      *PARAMETER  =  *87 (oder wenn einer der Teilwerte übereinstimmt,
                             wenn *PARAMETER oder *87 Teilwertlisten sind).
oder      *PARAMETER  =  MORITZ
oder      *PARAMETER  =  MAX
oder      BV1 = TRUE
oder      BV5 = TRUE ist.
```

Sind nun die Wahlschalter gesetzt worden, so wird

*** JA *** gemeldet , sonst *** NEIN *** .

WAHLSCHALTER

Spez.-Wert

-	Es wird nichts verändert, sondern nur evtl. eine Meldung gedruckt.
WS1 ... WS8	Wahlschalter 1 bis 8
ZW2 ... ZW5, ZW7, ZW9 ... ZW16	die angegebenen Zustandswahlschalter
FE1, FE2	die beiden Fehlervariablen des Entschlüsslers
ENDE	in der Grundstufe: Abbruch des Abschnitts in der Vorrangstufe: Rückkehr in Grundstufe
NEGIER[EN]	umkehrung der Aktion
LOESCH[EN]	evtl. Löschen der Wahlschalter, aber auf keinen Fall setzen
SETZEN	evtl. Setzen der Wahlschalter, aber auf keinen Fall löschen.

optionale Spezifikation zum Kommando ZUSTAND

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Diese Spezifikation steuert die Aktion des Operators, abhängig davon, ob der abgeprüfte Zustand vorliegt oder nicht. Der Wert'undefiniert' ist nur sinnvoll bei MELDUNG ≠ 'undefiniert', da dann nur geprüft wird, ohne irgend etwas zu verändern. Bei den Werten FE1 und FE2 wird der Operatorlauf mit Fehlermeldung beendet, sodaß die Variablen FE1 und FE2 gesetzt sind (für Kommando FEHLERHALT etc.)

Bei ENDE wird der Operatorlauf mit Fehlerschlüssel '3' beendet, d.h. sofortige Rückkehr aus der Vorrangstufe bzw. in der Grundstufe Abbruch des Abschnitts.

Normalerweise werden die angegebenen Wahlschalter bei erfüllter Bedingung (siehe 'FRAGE') gesetzt, sonst gelöscht. Ist NEGIER als Teilwert angegeben, so werden die Wahlschalter bei erfüllter Bedingung gelöscht, sonst gesetzt.

Ist LOESCH angegeben, so werden die Wahlschalter zwar evtl. gelöscht, ein sonst gefordertes Setzen aber unterbleibt, bei SETZEN entsprechend umgekehrt.

Für die Spezifikationswerte FE1, FE2, ENDE, NEGIER, LOESCH, SETZEN genügt auch der erste Buchstabe.

2. 4. Dez. 1972

formal:

⟨Wertzuweisung Wahlschalter⟩ ::= [WAHLSCHALTER =] { ⟨Teilwert⟩['⟨Teilwert⟩] }∞

⟨Teilwert⟩ ::= WS { 1 2 3 4 5 6 7 8 } | ZW { 2 3 4 5 7 9 : : 16 } | { F[E] F[E] } | L[OESCH[EN]] | S[ETZEN] | N[EGIER[EN]] | E[NDE]

Beispiel:

...,WAHL.=ZW4,F.= -STD-,...
Der Zustandswahlschalter 4 wird auf jeden Fall gesetzt.

...,WAHLSCHALTER=WS1'WS6,...
Die Wahlschalter WS1 und WS6 werden gesetzt, wenn die Frage nach dem Zustand mit ja zu beantworten ist, sonst gelöscht.

...,WAHL.=ENDE,FRAGE=VOR,...
In Grundstufe keine Aktion, eine evtl. Vorrangstufe wird sofort beendet (allerdings mit einer unschönen Fehlermeldung).

...,WA.=ZW5'NEGIER,FRAGE=ZW5,...
Ist der Zustandswahlschalter 5 gesetzt, so wird er gelöscht, ist er gelöscht, so wird er gesetzt.

```
ZUST., W.=WS5,F.=BKZ,V.=MORITZ
ZUST., W.=WS5'LOESCH,F.=FKZ,V.=MAX
```

Diese beiden Kommandos bewirken, daß der Wahlschalter WS5 gesetzt wird, wenn BKZ=MORITZ und FKZ=MAX sind(Wenn unter 'FRAGE' sonst zwei Teilwerte angegeben sind, werden sie durch das logische 'OR' verknüpft.).

FRAGE

	-	Bedingung nie erfüllt.
	-STD-	Bedingung immer erfüllt.
	WS1...WS8	ob einer der Wahlschalter gesetzt ist,
Spez.-Wert	BV1...BV8	ob eine der booleschen Variablen den Wert TRUE hat,
	ZW1...ZW16	ob einer der Zustandswahlschalter gesetzt ist,
	SI1...SI24	ob eines der Signale gesetzt ist,
	GSP	ob ein Gespräch vorliegt,
	SIG	ob am Sichtgerät gearbeitet wird,
	VOR	ob Vorrangstufe herrscht,
	EINGAB	ob eine Anfrage, die im Gespräch an der Konsole gestellt wird, mit JA beantwortet wird - im Abschnitt immer NEIN.
	PRUEF	ob irgendein Teilwert von 'VERGLEICH' als Teilwert in 'PRUEFEN' enthalten ist.
	FST	ob unter 'PRUEFEN' ein Gebietsfremdstring angegeben ist.
	UNDEF	ob 'PRUEFEN' den Wert 'undefiniert' hat.
	STD	ob 'PRUEFEN' den Wert -STD- hat.
	BKZ	ob eines der 4 eingetragenen BKZ's mit der Zeichenfolge unter 'VERGLEICH' identisch ist.
	FKZ	ob das FKZ des Abschnitts mit der Zeichenfolge unter 'VERGLEICH' identisch ist. Blanks im FKZ sind unter 'VERGLEICH' als Ausrufezeichen anzugeben. Ist 'VERGLEICH'='undefiniert', so wird mit 'JA' geantwortet, wenn kein FKZ angegeben ist.
	DB	ob die Datenbasis, die unter 'VERGLEICH' angegeben ist, existiert.
	LISTE	ob unter 'PRUEFEN' eine Liste von Teilwerten angegeben ist,
		ob einer der folgenden Werte größer oder gleich dem unter 'BEDARF' angegebenen ist:
	SBG	Speicherbedarfsgruppe
	KSB	Kernspeicherbedarf
	TSB	Trommelspeicherbedarf
	PSB	Plattenspeicherbedarf
	BGB	Bandgerätebedarf
	RZS	Rechenzeitschranke (in Sekunden!)
	DRS	Druckseitenschranke
	RTSP	Restlicher zur Verfügung stehender Trommelspeicher
	RPSP	Restlicher zur Verfügung stehender Plattenspeicher

Diese Bedarfswerte werden von rechts nach links den unter 'BEDARF' angegebenen Zahlen zugeordnet, die in gleicher Anzahl vorhanden sein müssen.

optionale Spezifikation zum Kommando ZUSTAND	anlagenspezifische Voreinstellung: "undefiniert"

Wirkung:

Mit dieser Spezifikation wird gesteuert, welcher Zustand untersucht werden soll. Eine geforderte Meldung bezieht sich immer auf den Zustand, d.h. ob ***JA(SPEZIF)*** oder ***NEIN*** gemeldet wird. Mehrere, durch Apostrophe getrennte Angaben werden nacheinander von rechts nach links abgearbeitet, und bei der ersten zu bejahenden Überprüfung wird der Operator beendet (evtl. mit entsprechender Meldung). Die dabei ausgeführte Aktion hängt nur von 'WAHLSCHALTER' ab.

Sind die Spezifikationswerte BKZ,FKZ oder DB angegeben, so dürfen unter 'VERGLEICH' nicht mehrere Werte auftreten, d.h. auch diese 3 Spezifikationswerte dürfen nicht gemeinsam auftreten, alle anderen dürfen beliebig gemischt auftreten.

Ist Frage = EINGAB, so wird sofort alle bisher angesammelte Druckinformation ausgegeben und eine Eingabe angefordert:
FRAGE VON ZUSTAND ◘:
Bei nun eingegebenen vorrangigen Kommandos werden diese ausgeführt, danach wird die Anfrage wiederholt. Wird JA eingegeben, so gilt die Bedingung als erfüllt, bei leerer oder anderer Eingabe nicht.
Alle Spezifikationen können im Rahmen der Eindeutigkeit beliebig durch Punkt abgekürzt werden, z.B.: E.=EINGAB

Format

⟨Wertzuweisung FRAGE⟩ ::= [FRAGE=] { -STD- / ⟨Teilwert⟩ ['⟨Teilwert⟩]^∞ }

⟨Teilwert⟩ ::= | GSP | SIG | VOR | { WS / BV } { 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 } | FKZ | EINGAB | BKZ | DB |

ZW{ 1 : 16 } | SI{ 1 : 24 } | PRUEF | FST | UNDEF | STD | LISTE |
TSB | PSB | BGB | RZS | DRS | SBG | KSB | RTSP | RPSP

Beispiele

...,FRAGE=FKZ'G.,...,VERGLEICH=!TEST

Es wird untersucht, ob das FKZ (siehe XBA...,XBG...-Kommando) mit der Zeichenfolge ␣TEST identisch ist, oder ob Gesprächszustand herrscht.

ZUSTAND,W.=ZW4,FRAGE = -STD-
Es wird in jedem Fall der Zustandswahlschalter ZW4 gesetzt.

BEDARF

Spez.-Wert: n natürliche Zahl < 10000

optionale Spezifikation zum Kommando ZUSTAND	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Ist unter 'FRAGE' einer oder mehrere der Bedarfswerte des Abschnittskommandos angegeben ('SBG', 'BGB' etc.), so wird dieser mit der hier angegebenen Zahl verglichen und der abgeprüfte Zustand gilt als zutreffend, wenn der entsprechende Bedarfswert größer oder gleich der angegebenen Zahl ist.

(syntax)

⟨Wertzuweisung BEDARF⟩ ::= [BEDARF =] { ⟨Bedarfswert⟩['Bedarfswert]^∞ }

⟨Bedarfswert⟩ ::= ⟨Ziffer⟩[⟨Ziffer⟩]^3

(beispiel)

...,FRAGE=KSB'BGB'RP.,...,BEDARF=75'3'400

Es wird geprüft, ob KSB ≥ 75 oder BGB ≥ 3 oder der restliche zur Verfügung stehende Plattenspeicher ≥ 400 ist. Für KSB, BGB etc. siehe XBA ..., XBG ... - Kommando.

PRUEFEN

Spez.-Wert: Fremdstrings
und
Normalstrings

optionale Spezifikation zum Kommando ZUSTAND	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Im allgemeinen wird man hier formale Parameter einer Kommandoprozedur angeben, deren jeweiliger aktueller Wert gemäß 'FRAGE' untersucht werden soll, z.B. ob der aktuelle Wert "undefiniert" (FRAGE=UNDEF) oder -STD- (FRAGE=STD) ist, oder es wird auf Identität eines Teilwertes mit einem der unter 'VERGLEICH' aufgeführten Teilwerte abgeprüft (FRAGE = PRUEF) etc. Es können unter 'PRUEFEN' auch mehrere formale Prozedurparameter angegeben werden, durch Apostroph getrennt, die untersucht werden sollen.

Ist z.B. FRAGE = FST, so wird jeder der aktuellen Parameter untersucht, und ist einer davon ein Gebietsfremdstring, so wird die geforderte Zustandsuntersuchung mit JA beantwortet. Die Werte "undefiniert" und -STD- dürfen nicht als Teilwerte auftreten! Ist Frage = FKZ und PRUEFEN = "undefiniert", so wird der Wahlschalter gesetzt, wenn im Abschnittskommando kein FKZ angegeben ist.

ZUSTAND / PRUEFEN

Format

⟨Wertzuweisung PRUEFEN⟩ ::= [PRUEFEN =] { ⟨String⟩ [, ⟨String⟩]^∞ }

⟨String⟩ ::= { ⟨Fremdstring⟩ / ⟨Normalstring⟩ } [◇/]

⟨Fremdstring⟩ ::= Zeichenfolge, die kein ◇ enthält, außer in der Kombination ◇⟨Ziffer⟩³

⟨Normalstring⟩ ::= siehe Syntax der Kommandosprache 3.9

Beispiel

...,FRAGE = FST'LI.,...,PRUEFEN = *PARAMETER3,...

Es wird untersucht, ob der formale Prozedurparameter *PARAMETER3 als aktuellen Wert eine Liste von Teilwerten oder einen Gebietsfremdstring hat.

VERGLEICH

Spez.-Wert: Fremdstrings
und
Normalstrings

optionale Spezifikation zum Kommando ZUSTAND	anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Ist FRAGE = PRUEF, so werden die Wahlschalter der 1. Spezifikation gesetzt, wenn einer der Teilwerte von 'VERGLEICH' mit einem der Teilwerte von 'PRÜFEN' (siehe dort) identisch ist. Gebietsfremdstrings werden dabei nicht auf Identität geprüft.

Ist FRAGE = FKZ bzw. BKZ, so wird das FKZ bzw. die BKZ's (im Normalfall sind dies das eigene BKZ + "KFD") mit untersucht, ob es bzw. eines von ihnen mit der unter 'VERGLEICH' angegebenen Zeichenfolge übereinstimmen. Enthält das FKZ Leerzeichen (außer den Leerzeichen am Ende), so müssen an deren Stelle unter 'VERGLEICH' Ausrufezeichen angegeben werden.

Format

⟨Wertzuweisung VERGLEICH⟩ ::= [VERGLEICH =] { ⟨string⟩ [⟨string⟩]^∞ }

⟨string⟩ siehe Spezifikation PRUEFEN

Beispiel

...,FRAGE=PRUEF,...,PRUEFEN=*87,...,VERGLEICH=MB(111222),...

Es wird untersucht, ob der interne Name *87 als Wert oder als Teilwert die Zeichenfolge MB(111222) hat.

MELDUNG

Spez.-Wert:

-	keine Meldung
-STD-	Ist die Frage nach einem bestimmten Zustand aufgrund der aktuellen Situation mit Ja zu beantworten, so erscheint bei eingeschaltetem Druckerprotokoll die Meldung JA, sonst die Meldung NEIN.
KO	wie -STD- mit zusätzlicher Ausgabe auf Konsole.
SOFORT	wie KO mit sofortiger Ausgabe, d.h. Verdrängung des Gesprächs.

optionale Spezifikation zum Kommando ZUSTAND

anlagenspezifische Voreinstellung: "undefiniert"

Einschränkung:

Wirkung:

Protokollierung auf Drucker bzw. bei MELDUNG = KØ auch auf der Konsole, ob die Frage nach dem Zustand mit "JA" oder mit "NEIN" zu beantworten war - also auch wenn unter 'WAHLSCHALTER' der Teilwert'NEGIER' angegeben war, wird bei erfüllter Zustandsbedingung ein JA ausgegeben, obwohl dann die Wahlschalter gelöscht wurden.

Wird ein JA ausgegeben, so erscheint dahinter in Klammern der Spezifikationswert von 'FRAGE', der dieses "Bejahen' bewirkt hat, z.B.: ***JA(EINGAB)***, sonst wird ***NEIN*** gedruckt. Bei eingeschaltetem Druckerprotokoll erscheint die Meldung auch im Gespräch auf dem Schnelldrucker.

24. DEZ. 1972

Format

⟨Wertzuweisung MELDUNG⟩ ::= [MELDUNG=] { ⟨Meldungsart⟩ }

Meldungsart ::= -STD- | KØ | SOFORT

Beispiel

...,M.=SOFORT,...

a) im Gespräch:

Es erfolgt sofort eine Ausgabe aller bisher angesammelten Druckinformationen (unter gleichzeitiger Verdrängung des Gesprächs) mit Meldung auf der Konsole, ob die angegebenen Wahlschalter aufgrund der Situation und der Frage gesetzt wurden oder nicht.

b) im Abschnitt:

Wirkung wie MELDUNG = -STD- oder KØ, d.h. einfach eine Meldung auf dem Schnelldrucker, ob der geprüfte Zustand vorliegt.

ZUST.,W.=-, FRAGE = S I 10,...,MEL.=KØ

Es wird kein Wahlschalter verändert, es wird nur JA(S I 10) oder NEIN auf der Konsole gedruckt - je nachdem, ob das Signal 10 gesetzt ist oder nicht.

Bisher erschienene Arbeitsberichte des Rechenzentrums

der Ruhr-Universität Bochum

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

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

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

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

Nr. 7105: H. Wupper
WR 1802 - Ein einfaches Band-Betriebssystem für einen mittleren Rechner

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Nr. 7304: M. Hauenschild
Ansätze zur komplexen Kreisarithmetik

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Nr. 7603: R. Buchmann, M. Rosendahl
BOGOL-TAS, eine Spracherweiterung von ALGOL60 durch Codeprozeduren zur Systemprogrammierung

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

Nr. 7605: 8. Jahresbericht des Rechenzentrums (1. Januar bis 31. Dezember 1975)

Nr. 7606: R. Buchmann, H. Wupper
BO&ZEICHNE, Vorschlag zur geräteneutralen Graphik

Nr. 7701: Camici, Ehlich, Kipp, Wiedemann
Inventarisierungsprogramm

Nr. 7702: K.A. Görg, W. Stark, R. Wojcieszynski
Verfahren zur Berechnung der Strömung durch eine Drosselstelle ohne Speicherwirkung

Nr. 7703: H. Wupper
ALGOL68 Eine Einführung in die Programmierung für Anfänger

Nr. 7704: Camici, Prof. Dr. Ehlich, Volmer, Wiedemann
Aufbau und Verwendung der Personaldatei der Ruhr-Universität Bochum

Nr. 7705: M. Rosendahl
AIDA, Handbuch für den Benutzer

Nr. 7706: V. Riedel, K.-H. Schloßer
Ein Programmsystem zur Ausleihverbuchung und interaktiven Rechnerunterstützung in der allgemeinen Buchbestandsverwaltung
Teil 2: On-line-System

Nr. 7707: M. Peuser, H. Wupper
Ein geräteneutrales und rechnerneutrales System zur graphischen Ausgabe

Nr. 7708: H. Wupper
ERZEUGE, eine Erweiterung der Kommandosprache

Nr. 7709: A. Heidt, G. May
ADWAND ein System zur Bearbeitung von Analogdaten an TR86 und TR440

Nr. 7710: R. Buchmann
BO&PAGING Der virtuelle Kernspeicher für den TR440

Nr. 7711: R. Buchmann, H. Wupper
Unzulänglichkeiten des TR 440 Programmiersystems und ihre Umgehung
2.geänderte Auflage