SIEMENS

1463

# Datenverarbeitung Systeminformationen

data spezial

Ausgabe 1/1977

Herausgeber
Siemens AG, Bereich Daten- und Informationssysteme
Postfach 700078
Telefon (089) 722-22639

Anfragen und Manuskripte bitten wir zu richten an
Siemens AG, Bereich Daten- und Informationssysteme
Redaktion »data spezial – Systeminformationen«

Für Fragen zu einzelnen Beiträgen, soweit sie nicht von dem Systembetreuer der regional zuständigen Zweigniederlassungen beantwortet oder geklärt werden können, ist im allgemeinen der jeweils genannte Verfasser zuständig.

# Ergebnis der data spezial-Umfrage 2/1976

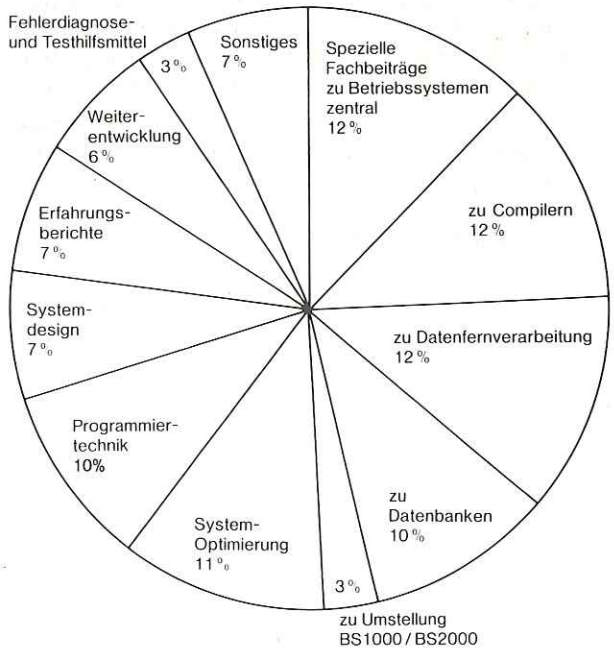
data spezial

Sehr geehrter data spezial-Leser,
die Redaktion bedankt sich für die lebhafte Beteiligung an ihrer Umfrage. Der Rücklauf liegt mit gut 20% weit über dem Durchschnitt (5%), wie er sich bei entsprechenden Umfragen bisher ergeben hat.

Wie sieht das Ergebnis aus?

1. Mehr als ein Drittel der Antworten stammen von Lesern, die data spezial nur gelegentlich oder selten in die Hand bekommen. Dies kann nur Ihr Systembetreuer im einzelnen prüfen und dann in der Empfängerliste berücksichtigen. Sofern es nicht an der Verteilung oder Weitergabe in Ihrem eigenen Unternehmen liegt.
2. An vier der sechs zum Test ausgewählten data spezial-Beiträgen wurde starkes Interesse bekundet. Dies bestätigt der Redaktion, in der Auswahl der veröffentlichten Beiträge im allgemeinen richtig zu liegen. Auf Konsequenzen zu Ihren eigenen Themenwünschen wird unter 6. noch eingegangen.
3. Mehr als die Hälfte der Leser bestätigen, daß ihnen die Art der Themen zusagt, fast 40% möchten sie allerdings noch mehr praxisbezogen haben. Mit der Länge der Beiträge ist die Mehrheit einverstanden. Wenn zwei Drittel unserer Leser sich wünschen, data spezial öfter zu bekommen, müssen wir diesen Ball eigentlich an sie zurückspielen. Denn Erfahrungen und Entwicklungen, über die es zu berichten gilt, werden hauptsächlich bei den Anwendern, also bei Ihnen gemacht. Je zahlreicher Ihre Berichte bei der Redaktion eingehen, desto öfter kann data spezial erscheinen.

   data spezial hat sich Ihnen einst als Forum für einen Erfahrungsaustausch zum Nutzen aller Siemens-DV-Anwender vorgestellt. Daran möchten wir Sie bei dieser Gelegenheit erinnern.
4. Daß data spezial von fast allen Lesern archiviert
5. und von vielen gleich nach Erhalt durchgesehen wird, unterstreicht nur das in den anderen Antworten bekundete Interesse.
6. Unter den zahlreichen Themenwünschen lassen sich einige Schwerpunkte erkennen, die wir in einer kleinen Graphik wiedergeben. Abgesehen von dem Bestreben der Redaktion, diese Schwerpunkte bei der Wahl von Themen zu berücksichtigen, soweit es um Berichte aus den Reihen der Siemens AG geht, könnte der Leserkreis selbst Anregungen für seine Beiträge aus diesem Frageergebnis schöpfen.




Schwerpunkte waren:
- ○ Spezielle Fachbeiträge zu Betriebs-, Sub- und Datenbanksystemen
- ○ Beiträge zu Systemoptimierungen (System-Tuning und -Monitoring)
- ○ Erläuterungen diverser Programmiertechniken
- ○ Systemdesign, d. h. Zusammenhänge der Systemkomponenten und Schnittstellen
- ○ Erfahrungsberichte zum Einsatz von System- und Datenbanksoftware

7. Auch der letzte Fragenkomplex, der Aufschluß über die Bewertung von Informationsquellen gegeben hat, zeigt, daß spezielle Firmeninformationen allen anderen Quellen vorgezogen werden, also auch ein Pluspunkt für data spezial.

Erfahrungsaustausch, Fach- und Firmenzeitschriften sowie Fachberater werden ebenfalls gern genutzt. Vermutlich aufgrund zu geringer Teilnahmemöglichkeit stehen Kongresse, Seminare und Messen an letzter Stelle der Bewertung.

Sie erhalten hiermit die Ausgabe 1/1977 mit sechs Beiträgen. Ausgabe 2/1977 wird als Sonderheft mit einem einzelnen, sehr umfangreichen Beitrag über „Fehlerbehandlung im DVS BS2000" in Kürze folgen.

Wir hoffen, daß die Ausgabe 3/1977 wieder mit aktiver Beteiligung aus dem Leserkreis zusammengestellt werden kann.

Redaktion data spezial

# Dialogprogramme in ALGOL

von Jelske Kloppenburg
GMD Gesellschaft für Mathematik und Datenverarbeitung, Birlinghoven

data spezial

**Einleitung**

Um ein Dialogprogramm in ALGOL schreiben zu können, ist es nötig, die Arbeitsweise der Puffer-Ein-Ausgabe der ALGOL-Ein-Ausgabe-Prozeduren zu berücksichtigen.

**Deskriptoren**

ALGOL
Dialogprogramm
Puffer-Ein-Ausgabe

Eine der Möglichkeiten, einen Dialog mit Hilfe von ALGOL-Hilfsprozeduren zu verwirklichen, zeigen folgende Ausführungen.

In einem Dialogprogramm dient die Datensichtstation zugleich für Ein- und Ausgabe. Das sind im BS2000 SYDATA und SYSOUT. Diese beiden logischen Files werden von BS2000 ALGOL60 über die DSN's 3 und 4 angesprochen. Da bei ALGOL60 ein Aufruf einer Ein- oder Ausgabeprozedur nicht jedesmal eine direkte Ein- oder Ausgabe bewirkt, kann man ein Dialogprogramm natürlich nicht einfach durch abwechselnde Ein- und Ausgabe programmieren.

Man muß also, wenn der Text sofort auf das Datensichtgerät ausgegeben werden soll, einen 'Vorschub' bewirken. Das kann man im Format angeben oder durch SYSACT(4,14,1) erreichen. Bei der Eingabe verhält es sich ähnlich. Ein INSYMBOL verarbeitet das nächste Zeichen der Eingabe, dies kann aber das nächste, noch nicht abgearbeitete Zeichen der vorigen Eingabe sein. Man muß also auch hier vorher einen Vorschub bewirken, damit eine neue Eingabe vom DSG gelesen wird.

Die richtige Reihenfolge im Dialogprogramm ist also:
Output auf DSN 4
neue Zeile DSN 4
neue Zeile DSN 3
Input von DSN 3

Dabei ist zu beachten, daß vor dem ersten Input, wenn DSN 3 noch nicht geöffnet ist, kein Vorschub gegeben werden darf.

Diesen Wechsel von Aus- nach Eingabe besorgt die Prozedur FLIP:

```
'PROC' FLIP;
  'BEGIN''INT' I;
   SYSACT(4,14,1);
   SYSACT(3,11,I);'IF' I = 1 'THEN' SYSACT(3,14,1);
'END';
```

Zwei weitere Prozeduren, die das Programmieren von Dialogprogrammen erleichtern, sollen als Anwendungsbeispiel dienen:

```
'BOOL''PROC' FRAGB(S); 'STRING' S;
   'BEGIN''INT' I,J; 'PROC' FLIP; 'CODE' ALGLIB;
      'FOR' J:=1,J+1 'WHILE' J<5 'AND' I=0 'DO'
      'BEGIN' OUTSTRING(4,S); FLIP; INSYMBOL(3,'('JN')',I);
         'IF' I=0 'THEN'
          'BEGIN'OUTSTRING(4,'('J / JA  ODER  N / NEIN')');
             SYSACT(4,14,1)
          'END'
      'END';
   FRAGB:= I=1
'END';
```

```
'INT''PROC' FRAGI(S); 'STRING' S;
'BEGIN''INT' I; 'PROC' FLIP; 'CODE' ALGLIB;
    OUTSTRING(4,S); FLIP; ININTEGER(3,I); FRAGI:=I
'END';
```

Ein Demonstrationsprogramm mit einer beliebigen Prozedur SAMPLE könnte dann so aussehen:

```
'BEG' 'INT I, S;
  'PROC' OUTPUT; 'CODE';
  'INT''PROC' FRAGI; 'CODE' ALGLIB;
  'BOOL''PROC' FRAGB; 'CODE' ALGLIB;
  'INT''PROC' SAMPLE; 'CODE' ALGLIB;
  SYSACT(4,6,52);
  OUTSTRING(4,'('DIE PROZEDUR SAMPLE WIRD GETESTET')');
  SYSACT(4,14,1);
  'FOR' I:=1,I+1 'WHILE' FRAGB('('NOCHMAL ?')') 'DO'
    'BEGIN'
       S:=SAMPLE(FRAGI('('BITTE EINE ZAHL')'));
       OUTPUT(4,'(''('DAS ERGEBNIS IST ')'ZZD/')',S);
    'END';
  OUTPUT(4,'(''('ENDE DES PROGRAMMS')'/')')
'END';
```

# Checkpoint-Restart-Routine (CKPTIN)

von Helena Geißler
Institut für Medizinische Datenverarbeitung
der Gesellschaft für Strahlen- und Umweltforschung mbH,
München

data spezial

**Inhalt**



**Deskriptoren**

Batch-Job
Checkpoint-Einsprung
Checkpoint-Unterbrechung
Restart-Routine

## 1. Einleitung

Es hat sich immer wieder erwiesen, wie notwendig eine Unterbrechungsroutine für Batch-Jobs ist. Die GSF bietet dafür eine Checkpoint-Restart-Routine (CKPTIN)* an.

Das Programm CKPTIN ermöglicht es dem Operateur, jeden beliebigen (Langläufer-)Batch-Job mit /INTR abzubrechen und durch /RESTART zu jedem beliebigen späteren Zeitpunkt an der Abbruchstelle wieder zu aktivieren. Allerdings ist CKPTIN für EAM-Files nicht anwendbar. Auch die Verwendung von Banddateien ist nur bedingt möglich (Beispiel 4).

Das Programm CKPTIN ist in der Assemblersprache geschrieben und als Subroutine in übergeordneten Programmen (ASSEMBLER, COBOL, FORTRAN) zu verwenden. Es enthält verschiedene Unterbrechungszweige und bietet daher differenzierte Anwendungsmöglichkeiten, die noch näher beschrieben werden.

Zwingend für die Anwendung einer Unterbrechung allerdings ist die Initialisierung der verschiedenen Unterbrechungszweige im Benutzerprogramm. Sie geschieht durch den erstmaligen Aufruf von CKPTIN, der zu Beginn des übergeordneten Programms stattfinden muß (Beispiele).

* Das Programm kann bei Einsendung einer Bandspule (800/1600 bpi) bei der GSF unter folgender Anschrift bezogen werden:
RZ der GSF, IMD, Arabellastraße 4, 8000 München 81
Telefon 089/91106 1–68.

## 2. Checkpoint-Unterbrechung

Erst nach der Initialisierung hat der Benutzer drei Möglichkeiten der Unterbrechung:

a) willkürliche Unterbrechung
   Checkpoint-Einsprung mit /INTR (Beispiel 1)
b) direkte Unterbrechung
   Checkpoint-Einsprung mit /INTR (Beispiel 2)
c) direkte Unterbrechung
   Checkpoint-Einsprung ohne /INTR (Beispiel 3)

Geschieht eine dieser drei Unterbrechungen, wird auf die entsprechenden, bereits initialisierten Unterbrechungsroutinen im Programm CKPTIN verzweigt. In jeder dieser drei Routinen wird ein Checkpoint geschrieben, d. h. ein Checkpoint- File eingerichtet, der Name dieses Files mit CHECKPOINT. VOM. <Datum, Uhrzeit> modifiziert und mit allen Informationen versorgt, die nötig sind, den abgebrochenen Job später mit /RESTART wieder an dieser Stelle zu aktivieren. Der Benutzer selbst braucht zum Checkpoint-File keinerlei Angaben zu machen.

Es besteht die Möglichkeit, einen Batch-Job mehrmals zu unterbrechen, d. h. mehrere Checkpoints zu schreiben. Im Programm CKPTIN ist aus Effektivitätsgründen (Platz, Geschwindigkeit) die Einrichtung nur **eines** Checkpoint-Files vorgesehen, so daß jede weitere Unterbrechung ein Updating desselben Files bewirkt, d. h. das Checkpoint- File enthält nach Abbruch eines Jobs nur die neuesten Informationen der zuletzt gültigen Unterbrechung, da die vorhergehenden bei jeder Unterbrechung überschrieben werden.

Dem Benutzer ist zu empfehlen, vorher den Operateur zu verständigen, falls ein (Langläufer-) Batch-Job mit CKPTIN arbeitet, d. h. daß dieser Job unterbrochen werden kann oder soll.

**3. Restart** (Beispiel 5)

Die Wieder- Aktivierung eines zu einem früheren Zeitpunkt abgebrochenen Batch-Jobs geschieht in der Restart-Routine des Programms CKPTIN. Sie bewirkt ein Zurückgreifen auf das vorher beschriebene Checkpoint-File, das mit den darin enthaltenen Informationen sofort den Zustand zum Zeitpunkt des Programmabbruchs wiederherstellt. Der Benutzer darf deshalb keine weiteren Angaben machen außer denen, die im /RESTART-Kommando enthalten sind (z. B. keine File-Angaben).

Auch nach der Wieder-Aktivierung eines abgebrochenen Jobs läßt das Programm CKPTIN weitere Unterbrechungen zu. Der Vorgang des Abbruchs bzw. der Unterbrechung und des Wiederaktivierens ist beliebig oft wiederholbar.

Zu beachten ist, daß das Checkpoint-File, falls es nicht mehr benötigt wird, wieder gelöscht werden soll.

**4. Programmaufruf:**

Initialisierung im übergeordneten Programm

.
.
.

IPARM=0 zwingend für Initialisierung aller drei Unterbrechungs-Routinen
ISTXIT=0 für willkürliche Unterbrechung, Checkpoint-Einsprung mit /INTR

oder
ISTXIT=1 für direkte Unterbrechung, Checkpoint-Einsprung mit /INTR

oder
ISTXIT=2 für direkte Unterbrechung, Checkpoint-Einsprung ohne /INTR

CALL CKPTIN (IPARM, ISTXIT)
.
.
.

**5. Anfragen und Antworten:**

**Beispiel 1**

Willkürliche Unterbrechung (Unterbrechungs-Routine INTR) Checkpoint-Einsprung mit /INTR.

Diese Unterbrechung kann nur vom Operateur vorgenommen werden.

```
Programm:   IPARM=O
            ISTXIT=O                          } Initialisierung
            CALL CKPTIN (IPARM,ISTXIT)
            .
            .
            .
Console:  →/INTR
            *** CHECKPOINT *** WAS WOLLEN SIE?? (R/T/C/H)
          →C
            *** CHECKPOINT TAKEN ***
            CKPT: <Datum> <Uhrzeit> CHECK1; HFPG = <mm>
            *** CHECKPOINT *** WAS WOLLEN SIE??? (R/T)
          →T                                  Abbruch mit Checkpoint
oder      →R                                  Resume
            /INTR
            *** CHECKPOINT *** WAS WOLLEN SIE? (R/T/C/H)
            .
            .
            .
```

Antwortmöglichkeiten: R=Resume T=Term C=Checkpoint H=Help

**Beispiel 2**

Direkte Unterbrechung (Unterbrechungs-Routine CR)
Checkpoint-Einsprung mit /INTR

Beim direkten Checkpoint-Einsprung gibt es eine Besonderheit: Sobald der Operateur /INTR gibt, wird im Programm CKPTIN der Wert eines Feldes (INT) auf 1 gesetzt und zum übergeordneten Programm zurückgesprungen: INT bzw. COMINT muß über die Common-Liste versorgt werden. Der Benutzer hat die Möglichkeit, an jeder beliebigen Stelle im übergeordneten Programm dieses Feld INT abzufragen und– falls vom Operateur auf der Console ein /INTR-Kommando gegeben wurde – entsprechend zum Checkpoint-Einsprung zu verzweigen, d. h. CKPTIN mit den entsprechenden Parametern zu versorgen und ein weiteres Mal aufzurufen.

```
Programm:    COMMON COMINT /INT/                    siehe Anmerkung
             .
             .
             IPARM=O
             ISTXIT=1                          }    Initialisierung
             CALL CKPTIN (IPARM,ISTXIT)
             .
             .
             .
Console:     →/INTR
             .
             .
Programm:    IF(INT.NE.1)    GO TO 1
             IPARM=1
             CALL CKPTIN (IPARM,ISTXIT)             direkter Checkpoint-Einsprung
             .
             .
             .
           1 .
             .
             .
Console:     ***CHECKPOINT TAKEN***
             CKPT: <Datum> <Uhrzeit> CHECK1; HFPG=<mn>
             ***CHECKPOINT***WAS WOLLEN SIE?? (R/T)
           →T                                       Abbruch
           →R                                       Resume
             .
             .
             .
```

Anmerkung zu COMMON: Ist das übergeordnete Programm in der Assemblersprache programmiert, ist die Commonbehandlung wie folgt:

```
COMINT    COM
INT       DC   F'O'
<Name>    CSECT
```

**Beispiel 3**

Direkte Unterbrechung (ohne Unterbrechungs-Routine)
Checkpoint-Einsprung ohne /INTR

Beim direkten Checkpoint-Einsprung hat der Operateur keinerlei Möglichkeit, den Job mit /INTR abzubrechen.
Nur der Benutzer selbst kann im übergeordneten Programm festlegen, an welcher Stelle ein Checkpoint erfolgen soll. Da auch in diesem Fall ein Updating des Checkpoint-Files gewährleistet ist, ergibt sich die Möglichkeit, an verschiedenen Stellen des Programms weitere Checkpoint-Einsprünge einzubauen.

```
Programm:   .
            .
            .
            IPARM=O
            ISTXIT=2                          }   Initialisierung
            CALL CKPTIN (IPARM,ISTXIT)
            .
            .
            .
            IPARM=1
            CALL CKPTIN (IPARM,ISTXIT)            1. direkter Checkpoint-
                                                     Einsprung
            .
            .
Console:    ***CHECKPOINT TAKEN***
            CKPT: <Datum> <Uhrzeit> CHECK1; HFPG = <mn>
            ***CHECKPOINT***WAS WOLLEN SIE?? (R/T)
          → R                                     Resume
Porgramm:   .
            .
            .
            IPARM=1
            CALL CKPTIN (IPARM,ISTXIT             2. direkter Checkpoint-
                                                     Einsprung
Concole:    ***CHECKPOINT TAKEN***
            CKPT: <Datum> <Uhrzeit> CHECK1; HFPG = <mn>
            ***CHECKPOINT***WAS WOLLEN SIE?? (R/T)
          → T                                     Abbruch
```

**Beispiel 4** (zu Banddateien)

Bei Banddateien kann nur ein direkter Checkpoint-Einsprung wie bei 2.c) verwendet werden. Ein ordnungsgemäßes Schließen, Rückspoolen und Wiedereröffnen der Banddateien für die Wiederaktivierung des Jobs durch /RESTART muß vom Benutzer gewährleistet werden.

Programm:

```
.
.
IPARM=O
ISTXIT=2                                  } Initialisierung
CALL CKPTIN (IPARM,ISTXIT)

.
.
.
IPARM=1
CALL CKPTIN (IPARM,ISTXIT)
```

Console:

```
***CHECKPOINT TAKEN***
CKPT: <Datum> <Uhrzeit> CHECK1; HFPG = <mn>
***CHECKPOINT***WAS WOLLEN SIE?? (R/T)
→R                                        zwingend!
```

Programm:

```
.
.
ENDFILE  <mn>
REWIND   <mn>

.
.
.
```

**Beispiel 5** (zu Restart)

Bevor der abgebrochene Job wieder aktiviert wird, muß der Operateur bzw. der Benutzer mit:

/FSTAT CHECKPOINT.VOM.

erst den vollen Namen des Checkpoint-Files ermitteln.

Die entsprechende Information über die Halfpage ist der Console-Meldung bzw. der Druckerliste zu entnehmen.
Auch nach /RESTART gibt es weitere Unterbrechungsmöglichkeiten.

```
→ /RESTART CHECKPOINT.VOM. <Datum> <Uhrzeit>, Halfpage
  ***RESTART TAKEN***
  <weitere Unterbrechungsmöglichkeiten>
→ /INTR
  ***CHECKPOINT***WAS WOLLEN SIE?? (R/T/C/H)   Beispiel
→ C                                            RESTART nach 2.a)
  ***CHECKPOINT TAKEN***
  CKPT: <Daum> <Uhrzeit> CHECK1; HFPG = <mn>
  ***CHECKPOINT***WAS WOLLEN SIE?? (R/T)
→ T
```

**Bemerkung:** Falls eine Ein/Ausgabe-Datei benutzt wurde, muß vor einem /RESTART die Datei erst kopiert werden.

# Der Klassentest in ANS-COBOL

von Heinz Kröger
Siemens AG, München, Bereich Anwenderprogramme

data spezial

**Zusammenfassung**

Bedingt durch
- den Lochkartencode, der eine Unterscheidung zwischen positiven und negativen Werten einerseits und einer Gruppe von Buchstaben andererseits nicht zuläßt,
- die Anordnung der Lochertastaturen, durch die gewisse Fehllochungen nur über die richtige Kombination von Picture-Beschreibung und Ablochart erkannt werden, und
- das Eingabesystem, von dem unabhängig von der Picture-Beschreibung des Karten-Datensatzes der Karteninhalt stets entpackt-dezimal abgespeichert wird,

sind bei Einsatz des Klassentests genaue Kenntnisse der drei genannten Punkte und vor allem ihrer Abhängigkeiten unerläßlich, um exakte Aussagen zu erzielen.
Die sich ergebenden Probleme und die zu treffenden Maßnahmen werden aufgezeigt.

**Inhalt**



**Deskriptoren**

Ablochorganisation
ANS-COBOL
Kartenfeld
Klassentest
PICTURE-Beschreibung

## 1. Bedeutung des Klassentests

Bei der Verarbeitung von Lochkarten ist das Erkennen von Ablochfehlern eine wesentliche Programmaufgabe.
Eines der wichtigsten Werkzeuge jeder Lese- und Prüfroutine ist der Klassentest. Mit ihm soll festgestellt werden, ob ein nach dem Einlesen im Arbeitsspeicher abgelegtes Kartenfeld rein numerisch und nicht alphabetisch oder alphabetisch und nicht numerisch ist.
Die Abfrage auf ein alphabetisches Feld hat geringe Bedeutung. In der überwiegenden Mehrzahl aller Fälle werden anstelle von alphabetischen Feldern alphanumerische verarbeitet. Ferner können und sollen solche Felder keiner arithmetischen Operation unterzogen werden. Liegt ein Ablochfehler vor, so entsteht beim Drucken zwar ein Schönheitsfehler, besondere Bedeutung hat er jedoch nicht.

Dafür ist der Klassentest für das Erkennen rein numerischer Felder und für das Erkennen von Ablochfehlern in ihnen um so bedeutungsvoller.

Vor der Verarbeitung numerischer Werte wie Preise, Buchungsbeträge, Posten der Lohn- und Gehaltsabrechnung usw. müssen Fehler, soweit wie irgend möglich, erkannt und abgewiesen werden. Hierfür ist der Klassentest ein in der Praxis am meisten eingesetztes Instrument.

Mängel und Maßnahmen zur Abhilfe sollen im weiteren aufgezeigt und verdeutlicht werden.

## 2. Ablegen eingelesener Kartenfelder

### 2.1

Kartenfelder werden in dem durch die FILE DESCRIPTION (FD) definierten Karten-Datensatz unabhängig von ihrer Feldbeschreibung durch die PICTURE-Klausel mit READ lochkartengetreu im EBCDI-Code abgelegt.

Gleichgültig, ob ein Feld (Element) des Karten-Datensatzes mit

PIC X(n), PIC 9(n), PIC S9(n)
PIC S9(n) COMP-3

beschrieben ist, der Wert eines z. B. 9stelligen Kartenfeldes

␣␣␣␣␣␣1 2 3

wird stets als

40 40 40 40 40 40 F1 F2 F3

abgespeichert.

Aus dieser Tatsache ergibt sich, daß die vier Abfragen des Klassentests in jedem Fall ein Zwitterergebnis bringen: sie erkennen solch ein Feld weder als numerisch noch als alphabetisch an.

**Tabelle 1**

| Abfrage | wahr | falsch |
|---|---|---|
| IF NUMERIC | | X |
| IF NOT NUMERIC | X | |
| IF ALPHABETIC | | X |
| IF NOT ALPHABETIC | X | |

**2.2**

Eine Ausnahme bilden solche Kartenfelder, die über alle Spalten hinweg numerisch gelocht sind, u. U. auch durch Lochen führender Nullen.

Ein 9stelliges Feld, das mit

0 0 0 0 0 0 1 2 3

abgelocht und dessen Ablagefeld im Karten-Datensatz mit

PIC X(9), PIC 9(9), PIC S9(9)

beschrieben ist, würde korrekt als numerisch und nicht alphabetisch erkannt werden.

**Tabelle 2**

| Abfrage | wahr | falsch |
|---|---|---|
| IF NUMERIC | X | |
| IF NOT NUMERIC | | X |
| IF ALPHABETIC | | X |
| IF NOT ALPHABETIC | X | |

Wäre dagegen das Ablagefeld mit

PIC S9(17) COMP-3

beschrieben, so ergäbe sich – logischerweise – wieder ein Zwitterergebnis: das Feld würde weder als numerisch noch als alphabetisch erkannt werden (siehe Tabelle 1).

**3. Ablochorganisation und Kartenaufbau**

Ablochorganisationen, die das Ablochen führender Nullen zwingend machen, sind aus naheliegenden Gründen als ungeeignet anzusehen.

Ferner sind Lochkartenfelder in aller Regel auf den aufzunehmenden Maximalwert ausgelegt. Dieser Maximalwert wird aber tatsächlich nur in Ausnahmefällen abzulochen sein. In der Regel sind kleinere Werte zu verarbeiten, so daß Leerspalten entstehen.

Der Punkt 2.2. soll daher als nicht relevant im weiteren nicht mehr diskutiert werden.

Der Vollständigkeit halber wird jedoch zuvor auf einige Besonderheiten hingewiesen.

**4. Besonderheiten**

**4.1**

Beschreibung des Ablagefeldes mit PIC X(n).

Mit Vorzeichen (Überlochung) gelochte rein numerische Werte, wie z. B.

0 0 0 1 2 3 $\overset{+}{4}$
0 0 0 1 2 3 $\overline{4}$

werden als nicht numerisch erkannt. Aufgrund des Lochkarten-Codes entspricht '$\overset{+}{4}$ ' dem Buchstaben 'D' und '$\overline{4}$' dem Buchstaben 'M'.

**4.2**

Beschreibung des Ablagefeldes mit PIC 9(n)

Es gilt das unter 4.1. Gesagte.

**4.3**

Beschreibung des Ablagefeldes mit PIC S9(n).

Befindet sich in einem sonst rein numerischen Feld an letzter Stelle ein Buchstabe von A–I oder von J–R, dann wird dieses Feld im ersten Fall als positiver Wert, im zweiten Fall als negativer Wert betrachtet.

Mit solchen Fehllochungen ist durchaus zu rechnen. Sie ergeben sich aus der Struktur des Lochkarten-Codes und der Anordnung der Lochertastatur.

Hierzu zwei Beispiele:

**4.3.1**

Gelocht werden sollte:

0 1 2 3

Bei der '3' wurde fälschlicherweise nicht mehr die numerische Umschalttaste gedrückt. Statt der '3' würde also ein 'O' gelocht. Das 'O' hat den EBCDI-Code 'D6', was vom Klassentest her gesehen und auch bei der Verarbeitung den negativen Wert

0 1 2 $\bar{6}$

ergibt.

**4.3.2**

Gelocht werden sollte:

0 0 0 7 8 2

Bei der '2' wurde die numerische Umschalttaste nicht mehr gedrückt. Statt der '2' wurde ein 'I' gelocht. Das 'I' hat den Code 'C9'. Der Klassentest betrachtet diesen Wert als

0 0 0 7 8 $\overset{+}{9}$

**4.4**

Bei den unter 4.1. und 4.2. genannten Beispielen werden korrekt gelochte Werte abgelehnt und bei den unter 4.3. genannten werden fehlerhaft gelochte Werte als korrekt durchgelassen. Ihre weitere Verarbeitung führt unbemerkt zu falschen Ergebnissen, da der algebraische Wert verändert wurde.

## 5. Ablochen numerischer Felder ohne führende Nullen

**5.1**

Für alle mit Leertaste oder in Verbindung mit der Programmkarte mit Tabulator übersprungenen Spalten werden in den Ablagefeldern (Elementen) des Karten-Datensatzes, wie oben schon erwähnt, Zwischenräume (SPACES) abgelegt.

Bevor solch ein Feld dem Klassentest unterzogen werden kann, ist die Umwandlung der SPACES in Nullen unabdingbar. Es würde sonst als nicht numerisch angesehen und abgewiesen werden.

**5.2**

Die Umwandlung durch die Übertragung des Feldes in einen Zwischenspeicher mit Hilfe des MOVE-Verbs vorzunehmen, ist nicht ratsam. Das Ergebnis kann nicht vorausgesagt werden.

Hierzu ein Beispiel:
Das numerische Feld eines Karten-Datensatzes, also das Sendefeld, sei mit PIC 9(6), das Empfangsfeld mit PIC S9(11) COMP-3 beschrieben.

Sendefeld.
Abgelocht: ␣␣A / 1 2
Gespeichert: 40 40 C1 61 F1 F2

Empfangsfeld nach
der Übertragung: 00 00 00 01 11 2F

Bei dieser Operation wird also grundsätzlich das niedrigwertige Halbbyte des entpackt-dezimalen Sendefeldes in das Empfangsfeld übertragen.

Daraus resultiert, daß

| | |
|---|---|
| die Buchstaben A–I | C1–C9 |
| die Buchstaben J–R | D1–D9 |
| die Buchstaben S–Z | E2–E9 |
| der Schrägstrich (/) | 61 |
| das Minuszeichen(–) | 60 |
| das kaufm. Und (&) | 50 |
| Zwischenraum (SPACE) | 40 |

als Ziffern abgelegt werden.

Die obigen offensichtlichen Ablochfehler würden nicht erkannt werden!

**5.3**

Für das Umwandeln von SPACE bietet sich vielmehr das EXAMINE-Verb an:

EXAMINE FELD REPLACING LEADING
SPACE BY ZERO.

**5.3.1**

SPACE ist dem alphanumerischen Zeichenvorrat zuzurechnen. Der Einsatz dieser figurativen Konstante erfordert, daß auch die Feldbeschreibung eine alphanumerische ist [PIC X(n)].

Dagegen kann ZERO bei jeder Feldbeschreibung eingesetzt werden.

**5.3.2**

In allen denjenigen Fällen jedoch, in denen Kommata und Dezimalstellen zu berücksichtigen sind, muß andererseits die Feldbeschreibung eine numerische sein, z. B. PIC 9(n)V99.
Denn bei einer später zu vollziehenden Übertragung eines als alphanumerisch beschriebenen Feldes [PIC X(n)] in ein mit Komma beschriebenes Feld [PIC 9(n)V99] würde das Sendefeld als ganzzahlig angesehen werden.

Beispiel:

| | |
|---|---|
| Sendefeld | PIC X(9). |
| Abgelocht | 9 1 2 3 5 0 (DM 9.123,50) |
| Gespeichert | 40 40 40 F9 F1 F2 F3 F5 F0 |
| Umgewandelt | F0 F0 F0 F9 F1 F2 F3 F5 F0 |
| Empfangsfeld: | PIC 9(7)V99. |
| Nach der Übertragung: | F0 F9 F1 F2 F3 F5 F0 F0 F0 (DM 912.350,00) |

Nach der Übertragung würden die beiden ersten Stellen in diesem Fall zwei Nullen, abgeschnitten und die beiden Stellen hinter dem Komma mit Nullen aufgefüllt werden, wodurch ein völlig anderer algebraischer Wert entsteht.

**5.3.3**

Wie später noch dargelegt werden wird, sind mit Ausnahme eines einzigen Falles zusätzlich zum Klassentest noch weitere Abfragen, z. B. solche des Vorzeichen- oder des Vergleichstests, notwendig.

Die Abfragen des Vorzeichentests

```
IF (NOT) POSITIVE . . .
IF (NOT) NEGATIVE . . .
```

erfordern ein als numerisch beschriebenes Feld.

**5.3.4**

Das Feld des Karten-Datensatzes muß also über die REDEFINES-Klausel zweifach definiert sein:

```
01 KARTENSATZ.
   02 KART1 PIC 9(6)V99.
   02 FILLER REDEFINES KART1.
      03 KART1-E PIC x(8).
```

Die Umwandlung kann nun durch folgende Anweisung geschehen:

```
EXAMINE KART1-E REPLACING
        LEADING SPACE BY"O".
```

**5.4**

Aber auch nach der Umwandlung der SPACES in Nullen ist der Klassentest nicht immer in der Lage, einwandfrei numerische Felder und Ablochfehler in ihnen zu erkennen. Je nach Beschreibung des Feldes müssen andere Abfragen mit herangezogen werden.

**5.4.1**

Ist bekannt, daß das Kartenfeld nur absolute Werte enthält (keine Überlochung am Feldende), und ist die Feldbeschreibung mit

```
PIC 9(n)
```

vorgenommen worden, dann liegt unter diesen beiden Bedingungen der einzige Fall vor, in dem mit der Abfrage

```
IF (NOT) NUMERIC . . .
IF (NOT) ALPHABETIC . . .
```

ein numerisches Feld oder ein Ablochfehler in ihm einwandfrei erkannt wird.

Denn würde beim Eintasten der letzten Ziffer des Kartenfeldes die numerische Umschalttaste nicht mehr gedrückt werden und ergäbe sich daraus eine positive bzw. negative Ziffer (siehe Tabelle 3), dann würde dieser Wert als nicht numerisch betrachtet werden (siehe 4.1 und 4.2).

Wurde die Feldbeschreibung dagegen mit

```
PIC S9(n)
```

vorgenommen, dann ergeben sich beim Fehleintasten der letzten Ziffer – Nichtdrücken der Umschalttaste – Konsequenzen, die der Tabelle 3 zu entnehmen sind.

**Tabelle 3**

| Ziffer wird zu | Zeichen | Zeichen wird als nicht numerisch erkannt |
|---|---|---|
| 0 (F0) | / (61) | ja |
| 1 (F1) | U (E4) | ja |
| 2 (F2) | I (C9) = $\overset{+}{9}$ | nein |
| 3 (F3) | O (D6) = $\overline{6}$ | nein |
| 4 (F4) | J (D1) = $\overline{1}$ | nein |
| 5 (F5) | K (D2) = $\overline{2}$ | nein |
| 6 (F6) | L (D3) = $\overline{3}$ | nein |
| 7 (F7) | M (D4) = $\overline{4}$ | nein |
| 8 (F8) | , (6B) | ja |
| 9 (F9) | . (4B) | ja |

Von den sechs durch den Klassentest nicht erkannten Buchstaben stellen fünf eine negative, einer eine positive Zahl dar. Wird zusätzlich zum Klassentest noch auf POSITIVE abgefragt, so werden fünf der möglichen sechs Felder erkannt und abgewiesen.

Der immer noch 'durchrutschende' sechste Fehler kann dadurch abgefangen werden, daß durch Redefinition des Feldes das letzte Zeichen isoliert und auf 'I' abgefragt wird.

Karten-Datensatz:

```
01  KARTENSATZ.
    02  KART1  PIC S9(6)V99.
    02  FILLER  REDEFINES  KART1.
        03  FILLER  PIC X(7).
        03  KART11  PIC X.
    02  FILLER  REDEFINES  KART1.
        03  KART1-E  PIC X(8).
```

Abfragen:

```
IF KART1 NUMERIC
 IF KART1 POSITIVE
  IF KART1 NOT = "I"
   GO TO KORREKT.
 GO TO FEHLER.
```

Eine hundertprozentige Fehlererkennung ist damit auch in diesem Fall erreicht.

Die zweite Redefinition von KART1 ist für die Umwandlung der SPACES in Nullen notwendig (siehe 5.3.1).

**5. 4. 2**

Ist bekannt, daß das Kartenfeld nur positive oder nur negative Werte (Überlochung am Feldende) enthalten darf, so würden diese Felder nicht als numerisch erkannt werden, wenn die Feldbeschreibung mit PIC 9 (n) vorgenommen wird (siehe 4.1 und 4.2).

**5. 4. 2. 1**

Im Kartenfeld werden nur positive Werte gelocht.

Karten-Datensatz:

```
01  Kartensatz.
    02  KART1  PIC  9 (6) V99.
    02  FILLER  REDEFINES  KART1.
        03  KART10  PIC  9 (7).
        03  KART11  PIC  X.
    02  FILLER  REDEFINES  KART1.
        03  KART1-E  PIC  X (8).
```

Abfragen:

```
IF KART10 NUMERIC
 IF KART11 > QUOTE
  IF KART11 NOT > "I"
   GO TO KORREKT.
 GO TO FEHLER.
```

Würde statt der Abfrage 'IF KART11 > QUOTE' die scheinbar plausiblere 'IF KART 11 NOT < "A"' angewendet, so würde $\overset{+}{0}$ als fehlerhaft abgewiesen werden. Denn der Code für A ist C1 und der für $\overset{+}{0}$ ist C0. Daraus ergibt sich, daß $\overset{+}{0}$ kleiner als A betrachtet wird. Die figurative Konstante QUOTE, für die der Compiler den Code für das Anführungszeichen generiert (7F), ist in der Code-Tabelle der nächst kleinere Wert gegenüber $\overset{+}{0}$.

**5. 4. 2. 2**

Im Kartenfeld werden nur negative Werte gelocht.
Karten-Datensatz wie unter 5.4.2.1.
Abfragen:

```
IF KART10 NUMERIC
 IF KART11 > "I"
  IF KART11 NOT > "R"
   GO TO KORREKT.
 GO TO FEHLER.
```

**5. 4. 2. 3**

Wird das Feld des Karten-Datensatzes mit PIC S9 (n) beschrieben, dann werden auch solche Felder als numerisch betrachtet, die versehentlich absolut abgelocht wurden.
Zusätzlicher Programmieraufwand wird für das Erkennen dieses Fehlertyps jedoch nicht verursacht. Denn bei negativ zu lochenden Beträgen mündet die NEGATIVE-Abfrage, die ohnehin notwendig ist, bei Vorfinden eines absolut gelochten Wertes in den Falsch-Zweig und bei positiv zu lochenden Werten läßt die POSITIVE-Abfrage absolute Beträge richtigerweise zu.

Bei der Beschreibung des Karten-Datensatzes mit PIC S9 (n) ergeben sich gegenüber derjenigen mit PIC 9 (n) sowohl bei der Definition des Karten-Datensatzes als auch bei den Abfragen Vereinfachungen.

Beispiel für ein negativ zu lochendes Feld:
Karten-Datensatz:

```
01  KARTENSATZ.
    02  KART1  PIC  S9 (6) V99.
    02  FILLER  REDEFINES  KART1.
        03  KART1-E  PIC X (8).
```

Abfragen:

```
IF KART1  NUMERIC
 IF KART1  NEGATIVE
  GO TO  KORREKT.
GO TO  FEHLER.
```

Andere als der oben bezeichnete Ablochfehler bewirken entweder eine nicht zulässige Lochkombination oder ergeben einen nicht numerischen Wert. Sie werden bereits beim Einlesen oder später durch den Klassentest erkannt und abgewiesen.

**6. Schlußbetrachtung**

Das Erkennen fehlerhaft gelochter numerischer Werte ist wichtig.

Der Klassentest, oft noch sekundiert durch die ZERO-Abfrage, ist dazu nur dann in der Lage, wenn Ablochart und PICTURE-Beschreibung richtig kombiniert werden. In manchen Fällen sind zusätzliche Abfragen, vor allem solche des Vorzeichentests, unumgänglich.

Der geringste Programmieraufwand entsteht dann, wenn das Lochkartenfeld absolute Werte (keine Überlochung) enthält und die Feldbeschreibung des Karten-Datensatzes mit PIC 9 (n) vorgenommen wird. Nur in diesem Fall erzielt der Klassentest ohne zusätzliche Abfragen ein einwandfreies Ergebnis.

Enthält das Lochkartenfeld dagegen positive oder negative Werte, dann bringt die Feldbeschreibung mit PIC S9 (n) den geringsten Programmieraufwand.

Das Lochen negativer oder gar positiver Werte ist indessen keineswegs notwendig. Denn die Umwandlung absolut gelochter Werte in negative Beträge kann dem Programm überlassen werden.

So kann über die Kartenart, den Buchungsschlüssel oder andere Kennzeichen das Programm angestoßen werden, die Umwandlung vorzunehmen.

Wird dieser Weg beschritten, dann

- ist der Klassentest wirkungsvoll
- der Programmieraufwand gering
- das Ablochen weniger zeitaufwendig.

Selbstverständlich gibt es eine Reihe anderer Möglichkeiten für Plausibilitätskontrollen und auch andere Programmierstrategien.

Darum ging es nicht.

Hier sollten der engbegrenzte Begriff 'Klassentest' behandelt und die größtmögliche Effizienz aufgezeigt werden.

# RPG2 Systemergänzungen für BS1000

von Walter Holzer
Siemens AG, München, Bereich Datenverarbeitung

data spezial

**Zusammenfassung**

Es werden systemergänzende Unterroutinen für die Anwendung im BS1000 RPG2 vorgestellt und beschrieben.

**Deskriptoren**

BS1000
Unterprogramm
RPG

**Inhalt**



## 1. Allgemeines

Für die Anwendung in BS1000 RPG2-Programmen stehen folgende Unterroutinen zur Verfügung:

| Name | Funktion |
|---|---|
| #TIME | CPU-Zeit und Tageszeit |
| #ID | Identifikation von Betriebssystem, Zentraleinheit, Programmname, Befehlszähler |
| $DUMMY | Dummy-Datei |
| $CCRD | CCRD-Makro |
| $KONS | Konsole-Datei |

Die Unterroutinen laufen in BS1000 V 1.3 und V 1.4 ab und liegen als Modul sowie als Source vor. Sie werden von RPG2-Programmen über die Operation „EXIT" (1. Zeichen des Namens = #) bzw. über „SPECIAL-FILE-EXIT" (1. Zeichen des Namens = $) aktiviert.

Die Unterroutinen – ein Klasse 3-Produkt – sind als BS1000 Systemergänzung archiviert und zu beziehen bei

Siemens AG, Bereich Daten- und Informationssysteme, Programmdienst, unter der Bestell-Nr. P26484-E5006-A201/A202
Anschrift: Leopoldstraße 208, 8000 München 40, Telefon 089/3861/475

## 2. Bechreibung der Unterroutinen

Im folgenden wird eine detaillierte Anwenderbeschreibung der einzelnen RPG2-Unterroutinen gegeben.

### 2.1 #TIME (CPU-Zeit und Tageszeit)

Funktion:
Unterprogramm '#TIME' ermöglicht den Abruf der aufgelaufenen CPU-Zeit sowie der aktuellen Tageszeit in RPG2-Programmen.

Aufruf:
RPG2-Operation 'EXIT' mit Unterprogramm-Name '#TIME'.

CPU-Zeit und Tageszeit werden in folgenden, durch ULABL zu erklärenden Feldern hinterlegt:

| Feldname: | Länge (Bytes): | Inhalt: |
|---|---|---|
| @RTIME | 4 (gepackt) | CPU-Runtime in 1/1000 s |
| @TIME | 4 (gepackt) | Tageszeit in der Form OHHMMSS+ (+ = Vorz. X'C') |

Verknüpfung:
RPG2-Verknüpfungskonventionen.

Allgemein:
Unterprogramm '#TIME' wird als Modul hinterlegt und durch AUTOLINK mit dem rufenden RPG-Programm verbunden.

Mit jedem Aufruf 'EXIT #TIME' werden die oben genannten Felder in der Reihenfolge @ RTIME, @ TIME versorgt.
Es werden die EXEC-Makros RTIME und GETIM aktiviert. Die mit ULABL erklärten Felder müssen numerisches Datenformat, entpackte Länge 7 und 0 Dezimalstellen aufweisen. Es müssen nur die tatsächlich gebrauchten Felder als ULABL erklärt werden.
CPU-Zeit und Tageszeit sind in gepacktem Format vorhanden. Damit wird es möglich, mit diesen Daten im RPG2-Programm umgehend zu rechnen und sie für die Druckausgabe entsprechend mit Masken oder Schlüsseln aufzubereiten. Geeignete Druckmasken sind z. B.:
Für @ RTIME ␣␣␣0␣,␣␣␣' CPU-Zeit in Sekunden
@ TIME '0␣␣:␣␣:␣␣' Tageszeit HH:MM:SS

Modulgröße:
128 Bytes.

### 2.2 #ID (Identifikation von Betriebssystem, Zentraleinheit, Programmname und Befehlszähler)

Funktion:
Unterprogramm '#ID' kann zur Identifikation von Betriebssystem, Zentraleinheit, Programmname und Befehlszähler, unter welchen ein RPG2-Programm abläuft, verwendet werden.

Aufruf:
RPG2-Operation 'EXIT' mit Unterprogramm-Name '#ID'.

Die Identifikation wird in folgenden durch 'ULABL' zu erklärenden Feldern hinterlegt:

| Feldname: | Feldlänge (Bytes) | Inhalt |
|---|---|---|
| @ OSID | 10 (alphanum) | Betriebssystem |
| @ CPUID | 10 (alphanum) | Zentraleinheit |
| @ PNAME | 8 (alphanum) | Programmname |
| @ PC | 1 (alphanum) | Befehlszähler |

Die Feldinhalte sind linksbündig ausgerichtet.

Verknüpfung:
RPG2-Verknüpfungskonventionen.

Allgemein:
Unterprogramm '#ID' wird als Modul hinterlegt und durch Autolink mit dem rufenden RPG2-Programm verbunden. Mit dem Aufruf 'EXIT #ID' werden die obengenannten Felder versorgt.

Die mit 'ULABL' erklärten Felder im RPG2-Programm sind im Unterprogramm '#ID' definiert (ENTRY).
Es müssen nur die tatsächlich gebrauchten Felder als ULABL erklärt werden.
Es werden die EXEC-Makros OSID, CPUID und PNAME aktiviert.

Modulgröße:
104 Bytes.

### 2.3 $DUMMY (Dummy-Datei)

Funktion:
DUMMY File für RPG2-Ein-Ausgabe

Aufruf:
Special-File Exitname '$DUMMY' (Sp. 54–59, F-Karte).
Zulässige Dateiarten sind:
INPUT, OUTPUT, UPDATE.

Verknüpfung:
RPG2-Verknüpfungskonventionen.

Allgemein:
Unterprogramm $DUMMY wird als Modul hinterlegt und durch Autolink mit dem rufenden RPG2-Programm verbunden.
Mehrere verschiedene Aufrufe (F-Karten mit Dummy-File Exit) in einem RPG2-Objektprogramm sind zulässig.
Die Anwendung von Dummy-Files empfiehlt sich, wenn gewisse RPG2-Regeln erfüllt werden müssen, die für das Programm selbst eigentlich unnötig sind und lediglich Programmieraufwand und Speicherplatz erfordern.

Solche Regeln sind z. B.:
Mindestens eine Primary-Datei muß vorhanden sein
Mindestens eine Ein-Ausgabe-Datei muß vorhanden sein.

Dummy-Files können also Ein-Ausgabe-Dateien simulieren. Allerdings müssen minimale Ein-Ausgabe-Beschreibungen zur Erfüllung der Syntax vorhanden sein.

I-/O-Anweisungen enthalten sinnvollerweise keine Feldbeschreibungen für Dummy-Files.

Modulgröße:
32 Bytes.

Anwendungsbeispiele:

```
H
FDUMMY    UP   G          1                SPECIAL          $DUMMY
IDUMMY    AA   01
C                 'DUMMY-XY'DSPLY
C                           SETON                          LR
ODUMMY    D
```

### 2.4 $CCRD (CCRD-Makro)

Funktion:
Lesen von Daten- bzw. Programmsteuerkarten aus dem MONIT2-Bereich mit CCRD-Makro für RPG2-Programme.

Aufruf:
Special-File Exitname '$CCRD' (Sp. 54–59, F-Karte). Datei- und Eingabe-/Verarbeitungsanweisungen müssen syntaktisch und semantisch der Geräteart READER entsprechen. Geräteart (Sp. 40–46) ist 'SPECIAL'.

Verknüpfung:
RPG2-Verknüpfungskonventionen.

Allgemein:
Unterprogramm '$CCRD' wird als Modul hinterlegt und durch Autolink zum rufenden RPG2-Programm gebunden. Mehrere CCRD-Dateien hintereinander können nur durch Erklärung als Demand-File mit Operation 'READ' verarbeitet werden.
Das Lesen der Daten bzw. Parameter aus dem MONIT2-Bereich wird durch den Makro 'CCRD' realisiert. Beträgt die Satzlänge der CCRD-Datei im RPG2-Programm weniger als bzw. genau 80 Bytes, so wird die Übertragung der Daten in den RPG2-Eingabebereich immer in der im RPG2-Programm definierten Satzlänge ausgeführt.

Ist die Satzlänge größer 80 Bytes, dann wird jedoch nur in der Länge 80 übertragen.
Als Endekriterien im MONIT2-Bereich werden '// END', '/*' und '// START' interpretiert. Mit // START kann eine neue CCRD-Datei beginnen.

Parametrierung:
JOB CONTROL..... z.B.
// OPTION PARAM
// EXEC RPG2-Programm
.....
.....
..Daten/Parameter für 1. CCRD-Datei (MONIT2-Bereich) ..
.....
.....
// START
.....
.....
...2. CCRD-Datei....
.....
.....
// END
..
..

Fehler:
Bei Schreibbefehlen an die CCRD-Datei wird die Meldung '$CCRD – FALSCHE ANWENDUNG' ausgegeben, das Programm wird nach Schließen der Dateien beendet.

Modulgröße:
904 Bytes.

Anwendungsbeispiel:

```
H
FDUMMY   UP  G        1                SPECIAL        $DUMMY
FC1      ID  G       80                SPECIAL        $CCRD
FC2      ID  G       11                SPECIAL        $CCRD



IDUMMY   KF  99
IC1      KF  01   1 C1                                1  80 C1
I
I        KF  99
IC2      KF  02   1 C2                                1  11 C2
I
I        KF  99
C                          READ C1                                11
C    11                    READ C2                                LR
C    01        C1          DSPLY
C    02        C2          DSPLY
ODUMMY   D
```

### 2.5 $KONS (Konsole als Datei)

Funktion:
Konsole-Datei für RPG2-Programme.
$KONS erlaubt Ein-Ausgaben über Konsolen durch Geräteart 'SPECIAL' mit entspechenden Datei-Ein-Ausgabe-Beschreibungen.

Aufruf:
Special-File Exitname '$KONS' (Sp. 54–59, F-Karte).
Zulässige Einträge in der RPG2-F-Anweisung:
Dateiart (Sp. 15) – (I/O/U) INPUT/OUTPUT/UPDATE
Dateibezeichnung (Sp. 16) – (P/S/D) PRIMARY/SECONDARY/DEMAND
Dateiende (Sp. 17) – Wahlweise
Dateiformat (Sp. 19) – G (1 I/O-Area)
Blocklänge – Leer oder gleich Satzlänge
Satzlänge – Eingabe 72 B (max.)
Ausgabe 127 B (max.)
U-Indikatoren (Sp. 71–72) – Wahlweise.

Für I-Anweisungen bestehen keine Einschränkungen.
Als C-Operationen sind 'READ' und 'EXCPT' zulässig.
In den O-Anweisungen sind Zeile- bzw. Blattvorschubangaben unzulässig. Standard Vorschub auf Konsole ist 1 Zeile. Für mehrzeiligen Vorschub können Leerzeilen ausgegeben werden.
Erfüllen F-/I-/O-Anweisungen nicht die genannten Satzlängeneinschränkungen, können u. u. Daten verloren gehen.

Verknüpfung:
RPG2-Verknüpfungskonventionen.

Allgemein:
Unterprogramm '$KONS' wird als Modul hinterlegt und durch Autolink zum rufenden RPG2-Objektprogramm gebunden. Es können beliebig verschiedene Konsole-Dateien in einem RPG2-Programm vorhanden sein.

Unterprogramm '$KONS' überprüft die Satzlänge der Konsole-Datei im aktuellen FCB und begrenzt diese bei Dateiart INPUT/UPDATE auf 72 Bytes, bei OUTPUT auf 127 Bytes, wenn die für Blattschreiber maximal zulässigen Satzlängen überschritten werden. Die Ein-Ausgaben über Konsole erfolgen mit dem 'TYPE'-Makro des BS1000.

Die Anforderung einer Eingabe über Konsole an den Operateur geschieht durch die Zeichenfolge '«---'. Eingaben werden vom Operateur mit ETX-Taste beendet. Das logische Dateiende einer Konsole-Datei wird durch '/*'-Eingabe auf Konsole dargestellt. Dies veranlaßt das Setzen von 'LR' im RPG2-Programm.
Bei der Ausgabe von Daten über Konsole werden dem Text nachfolgende Leerstellen ausgeblendet. Dadurch wird Blank-Hämmern auf Blattschreibern vermieden.

Modulgröße:
200 Bytes.

Anwendungsbeispiel:

```
H
FDUMMY   UP  G        1              SPECIAL        $DUMMY

FKONSOLE UU  G       25              SPECIAL        $KONS
IDUMMY   KF  99
IKONSOLE KF  01
I                                                 1   5 FELD
C                        READ KONSOLE                    LR
C    01                  TESTN           FELD       U20304
C                        MOVE FELD       WIESE   50
C    02        WIESE     MULT 1,5
C    03        WIESE     MULT 2,5
C    01                  EXCPT
OKONSOLE E
O                   02       WIESE      10 '0 ,   '
O                   03       WIESE K    10
O                   04                  10 'LEEEEEEEEER'
```

# Datenschutz im BS2000

von Werner Herrmann
Siemens AG, Zweigniederlassung Köln

**Zusammenfassung**

Das BS2000 erlaubt im Multiprogramming bis zu 119 Benutzern die gleichzeitige Benutzung der Anlage zu verschiedensten Zwecken.
Es ist gleichzeitig Batch- und Dialogbetrieb möglich. Diese unterschiedlichen Betriebsarten in Verbindung mit der organisatorischen und räumlichen Trennung der verschiedenen Benutzer erforderten schon beim Design des Betriebssystems Maßnahmen zum Schutz der einzelnen Benutzer untereinander und der ihnen gehörenden Dateien. Darüberhinaus wurde es durch den streng modularen Aufbau des Systems ermöglicht, jederzeit Erweiterungen und Verbesserungen vorzunehmen.

In den folgenden Ausführungen wird unterschieden zwischen
- Programmen, die sich im Speicher befinden und ablaufen
- Dateien, in denen Daten und Programme gespeichert sind.

**Deskriptoren**

Benutzerkatalog
Benutzerkennung
BS2000
Dateikatalog
Datenschutz
Paßwortkontrolle
Privilegierter Modus
Speicherschutz
Systemverwalter

**Inhalt**



## 1. Schutz der Programme untereinander

Bei Programmen wird zwischen Klasse-I- und Klasse-II-Programmen unterschieden.

Klasse-I-Programme sind speicherresident und belegen stets einen zusammenhängenden Teil im Realspeicher. Der Schutz dieser Bereiche wird durch Benutzung des hardwareseitig eingebauten Speicherschutzes realisiert. Jedes Klasse-I-Programm hat einen eindeutigen 4-Bit-Speicherschutz-Schlüssel, der vor jedem Zugriff mit dem entsprechenden Eintrag im Schlüsselspeicher verglichen wird. Bei Gleichheit wird der Zugriff durchgeführt, bei Ungleichheit meldet die Hardware einen Fehler.

Da der jedem Programm zugeordnete Schlüssel in einer Systemtabelle und die Einträge im Schlüsselspeicher in einem für den Benutzer nicht adressierbaren Bereich stehen, ist es für das Benutzerprogramm unmöglich, Einträge irgendwo zu ändern.

Klasse-II-Programme sind seitenwechselbar. Zur Adressierung der im Realspeicher befindlichen Seiten wird die Adreßumsetzeinrichtung der Hardware benutzt. Die dazu erforderlichen Adressen und Tabelleneinträge sind im geschützten Bereich des Betriebssystems abgespeichert. Da die gesamte Seitenverwaltung von Hardware und Software übernommen wird, hat das Benutzerprogramm keinen

Zugriff zu Bereichen außerhalb seines eigenen Adreßraumes. Daher ist es dem Benutzerprogramm unmöglich, auf fremde Bereiche z. B. innerhalb des Betriebssystems oder eines anderen Benutzerprogramms zuzugreifen.

Bei jedem Versuch von Klasse-I- oder -II-Programmen, außerhalb des eigenen Bereiches zuzugreifen, wird das Programm durch die Hardware unterbrochen, und auch durch Wiederholungen ist kein Zugriff möglich.

### 2. Privilegierte Befehle und Makros

Während des Betriebes wird zwischen privilegiertem und nichtprivilegiertem Modus unterschieden. Benutzerprogramme arbeiten stets im nichtprivilegierten Modus, d. h. privilegierte Befehle werden von der Hardware und privilegierte Makros von der Software abgewiesen.

Ein Übergang in den privilegierten Modus (Erlauben von privilegierten Befehlen) ist nur durch Setzen eines Schalters in einem dem Benutzerprogramm nicht zugänglichen Register möglich. Vom Betriebssystem wird dieser Schalter beim eigenen Laden bearbeitet. Privilegierte Systemmakros werden vom System mit einem Fehlercode abgewiesen, wenn die Anwendung in den Systemtabellen nicht als erlaubt eingetragen ist. Sie können nur vom Systemverwalter unter dessen Kennung benutzt werden.

Ein Benutzerprogramm kann also weder privilegierte Befehle noch Makros ausführen und sich somit keinen unkontrollierbaren Zugang zum System, zur Hardware usw. verschaffen.

### 3. Stellung des Systemverwalters

Der Systemverwalter, der für alle dem System bekannten Benutzer Verwaltungsarbeiten übernimmt, hat Zugriff zu sämtlichen Betriebsmitteln im System. Er ist unter einer besonderen Kennung und Paßwort im Katalog eingetragen und ist als privilegierter Benutzer mit erweiterten Möglichkeiten zu sehen.

Bei der Ausführung von zentralen Verwaltungsarbeiten, wie z. B.

- Zulassen oder Sperren von Benutzern
- Vergabe oder Löschen von Kennungen

muß für ihn jederzeit die Möglichkeit bestehen, in den laufenden Betrieb einzugreifen.

Das Paßwort des Systemverwalters sollte häufig geändert werden, damit es nicht anderen Benutzern bekannt wird. Dadurch kann organisatorisch sichergestellt werden, daß nur ein definierter Personenkreis Zugang zu allen z. Z. im System befindlichen Daten hat.

### 4. Schutz des Benutzers

Jeder dem System bekannte Benutzer des BS2000 wird in einer zentralen Systemtabelle – dem Benutzer-Katalog – unter einer bestimmten Kennung, die nur dem Benutzer selbst und dem Systemverwalter bekannt ist, eingetragen. Zu den für die Sicherheit wichtigsten Daten in dieser Tabelle gehört das Paßwort, welches zur Überprüfung des Benutzers mit herangezogen wird.

Bei jedem „LOGON" wird vom System Kennung und Paßwort überprüft und so das Arbeiten im Rechner nur berechtigten Benutzern gestattet. Die gesamte Zeile des „LOGON-Kommandos" wird nach dem Abprüfen vom System durch mehrfaches Überschreiben unkenntlich gemacht. Bei Eingabe auf Bildschirm hat jeder Benutzer die Möglichkeit, den Schirm schon bei der Eingabe dunkel einzustellen bzw. er kann nach Abprüfung des Kommandos die Zeile oder den Bildschirm löschen.

Der Benutzer kann nicht Kennung und Paßwort – auch nicht sein eigenes – lesen. Bei Verlust muß in jedem Fall der Systemverwalter zur Auskunft eingeschaltet werden. Dieser hat jederzeit die Möglichkeit, Benutzerkennungen und Paßwörter zu lesen und zu ändern.

data_
spezial

### 5. Schutz der Benutzerdateien

Alle dem System bekannten Dateien sind unter bestimmten Benutzerkennungen im Dateikatalog eingetragen, d. h. sie sind benutzerorientiert katalogisiert. Nur der Benutzer, unter dessen Kennung die Datei katalogisiert ist, kann zu dieser Datei zugreifen.

Soll der Zugriff allen Benutzern möglich sein, kann die Datei vom Eigentümer als „sharable" erklärt werden. Soll der Zugriff nur einem begrenzten Benutzerkreis möglich sein, so ist die Datei mit Paßwörtern für

Lesen
Schreiben oder
Ausführen

zu versehen und diese sind den berechtigten Benutzern mitzuteilen.

Der Benutzer hat jederzeit die Möglichkeit, die Attribute seiner eigenen Dateien wie z. B. Name, Paßwörter und Erlaubnis der Mehrbenutzbarkeit zu ändern. Dies könnte durch organisatorische Maßnahmen gefordert werden, was eine weitere Steigerung der Dateisicherheit bringen würde.

### 6. Zugriffe zu Dateien

Zu Dateien kann auf logischer Ebene (vorhandene Zugriffsmethoden) nur von berechtigten Benutzern oder vom Systemverwalter zugegriffen werden. Physikalische Zugriffe zu Dateien sind im typischen BS2000-Betrieb (seitenwechselbare Programme) unmöglich, da die entsprechenden Makros für Klasse-II-Programme nicht erlaubt sind.

Aus Kompatibilitätsgründen ist es möglich, Programme speicherresident – Klasse I – ablaufen zu lassen. Die Anzahl der Speicherseiten für diese Programme und damit auch die generelle Lademöglichkeit kann beim Laden des Systems angegeben und später, falls erforderlich, geändert werden.

Von diesen Programmen kann auf physikalischer Ebene nur auf private, nicht jedoch auf öffentliche Datenträger zugegriffen werden, da jede symbolische Gerätenummer vom System abgeprüft wird.

### 7. Selbstladende Programme

Dateien und auch Datenträger werden im eigenen System vor unberechtigten Zugriffen geschützt. Wenn jedoch selbstladende Programme oder Hilfsprogramme anderer Betriebssysteme verwendet werden, ist ein Zugriff auf jede beliebige Datei oder jeden Datenträger möglich.

Derartige unberechtigte Zugriffe sind ausschließlich durch organisatorische Maßnahmen zu verhindern, wie z. B. Trennen der Datenträger verschiedener Betriebssysteme.

### 8. Zusammenfassung

Die im BS2000 implementierten Schutzmechanismen wie z. B.

Benutzerkatalog
Dateikatalog
Paßwortüberprüfung
Speicherschutz (real, virtuell)
Verbot physikalischer Zugriffe zu Dateien

garantieren allen Anwendern ein hohes Maß an Datenschutz.

# Maßnahmen zur Leistungsverbesserung der Katalogverwaltung im BS2000

von Michel Defamie
Software S.A., Namur

**Zusammenfassung**

Es werden Möglichkeiten aufgezeigt, durch geeignete Maßnahmen des Systemverwalters und Benutzers die Leistungsfähigkeit der Katalogverwaltung zu verbessern.

**Deskriptoren**

Benutzerkennung
BS2000
Katalogschutz
Katalogverwaltung
Leistungsverbesserung
Systemverwalter

## Einleitung

Die Kataloge von sieben BS2000-Benutzern wurden auf ihre Hauptmerkmale untersucht. Das Ergebnis ist in diesem Beitrag dargestellt mit dem Ziel, dem Systemverwalter wie auch dem Benutzer Hinweise zu geben, wie die Leistungsfähigkeit des Katalogsystems verbessert werden kann. Die folgenden Punkte werden untersucht und erläutert:

1. Anzahl der unter der Benutzerkennung TSOS katalogisierten Dateien
2. Anzahl der unter einer beliebigen Benutzerkennung katalogisierten Dateien; Anzahl der Benutzerkennungen im System
3. Sekundärspeicherzuordnung
4. Größe der Datei TSOSCAT
5. Länge der Dateinamen
6. Löschen von Dateien unter einer Benutzerkennung
7. Katalogschutz

Ausführliche programmiertechnische Erläuterungen sowie Benutzerschnittstelleninformationen sind in den unter „Literaturhinweise" angegebenen Manualen enthalten.

### 1. Anzahl der unter der Benutzerkennung TSOS katalogisierten Dateien:

Unter der Benutzerkennung TSOS sind drei Arten von Dateien katalogisiert:

A) Wichtige Systemdateien, welche vom System selbst benutzt und verwaltet werden (Anzahl etwa 20), z. B. JOINFILE, TSOSCAT, STARTUP, PAGINGAREA, usw.

B) Systemdateien, welche von allen Benutzern verwendet werden können (Anzahl etwa 100) z. B. ASSEMB, LMR, MACROLIB, EDT, usw.

C) Dateien unter der Benutzerkennung TSOS, welche vom Systemverwalter bzw. Operateur benutzt werden (Anzahl je nach RZ), z. B. Prozeduren, usw.

Im Betriebssystem BS2000 ist die Zugriffszeit zu einem bestimmten Katalogeintrag einer Datei sehr stark von der Anzahl der Dateien, welche unter einer Benutzerkennung katalogisiert sind, abhängig. Da jeder Benutzer Zugriff zu den unter B) genannten Systemdateien hat, ist die Zugriffszeit stark von der Anzahl der Dateien vom Typ C) abhängig. Die ermittelten Werte sind aus Tabelle1, Spalte 3, zu entnehmen.

**Je weniger Dateien unter der Benutzerkennung TSOS katalogisiert werden, um so besser wird der Durchsatz für jeden Benutzer.**

Es ist deshalb ratsam für den Systemverwalter, Dateien vom Typ C) unter einer besonderen Benutzerkennung für das Rechenzentrum zu katalogisieren (z. B. „OPERATOR", „STATISTIK", . . .).

**Jeder Benutzer verbessert den Zugriff zu den Systemdateien vom Typ B), wenn er $TSOS. vor dem Dateinamen angibt.**

z. B. /EXEC $TSOS.EDT (1)
oder /EXEC $EDT
wird schneller gefunden als:
/EXEC EDT (2)

Im Fall (2) werden zuerst die Katalogeinträge der entsprechenden Benutzerkennung nach der Datei EDT durchsucht und danach erst die Systemdateien unter TSOS.

Im Fall (1) werden nur die Systemdateien unter TSOS durchsucht.

Unter der Annahme, daß unter TSOS und der Kennung eines Benutzers die gleiche Anzahl von Dateien katalogisiert sind, wird der Zugriff im Fall (1) um die Hälfte verkürzt. Der Systemverwalter sollte darüber die Benutzer informieren.

### 2. Anzahl der unter einer beliebigen Benutzerkennung katalogisierten Dateien; Anzahl der Benutzerkennungen im System

Es wurde schon darauf hingewiesen, daß die Zugriffszeit zu einem Katalogeintrag von der Anzahl der unter einer Benutzerkennung katalogisierten Dateien abhängt. Deshalb kann man sagen:

**Den besten Systemdurchsatz erreicht ein Benutzer mit etwa 60 katalogisierten Dateien unter einer Benutzerkennung.**

Die ermittelten Werte sind aus Tabelle 1, Spalte 4, zu entnehmen.

In diesem Fall können zwei Verbesserungsmaßnahmen getroffen werden:

**a) vom Benutzer**

Der Benutzer sollte sorgfältig die Dateien auswählen, welche im gemeinschaftlichen Speicherbereich (public space) stehen müssen. Er sollte vermeiden, daß alte bzw. temporäre Dateien im gemeinschaftlichen Speicher bleiben.
Er sollte eine tägliche oder wöchentliche Bereinigung seiner „on-line"-Dateien in Verbindung mit einer Sicherung (TSOSMT/SYSUPD) oder dem Löschen von Dateien vornehmen.
Der Systemverwalter sollte die Benutzer zu dieser Handlungsweise anleiten.

**b) vom Systemverwalter**

Entsprechend dem obigen Hinweis sind vom Systemverwalter soviele verschiedene Benutzerkennungen wie möglich zu vergeben.
Falls ein Benutzer die Anzahl seiner Dateien nicht auf 60 begrenzen kann, sollte der Systemverwalter ihm vorschlagen, mehrere Benutzerkennungen (userids) einzuführen.

Als Beispiel wird folgender Zustand angenommen:

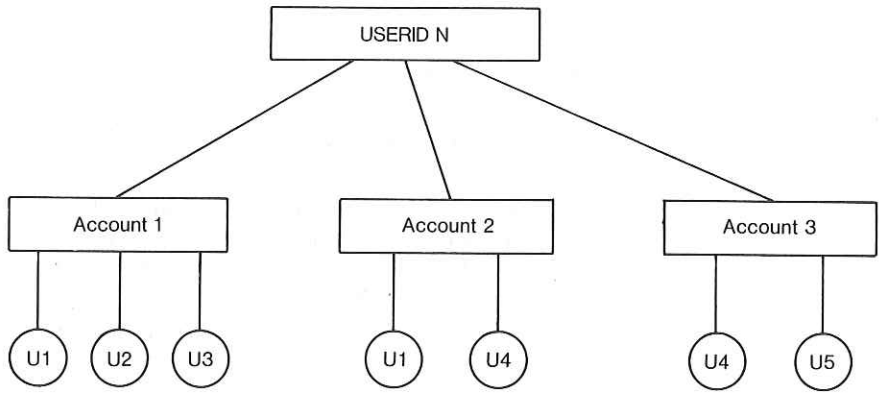



Bild 1

Anzahl der Dateien unter der Benutzerkennung N: etwa 450.

**Was kann getan werden, um einen optimalen Zustand zu erreichen?**

Die Benutzerkennung N wird aufgeteilt in mehrere Benutzerkennungen (userids), jeweils bezogen auf einen Benutzer mit mehreren Abrechnungsnummern (account nos.).

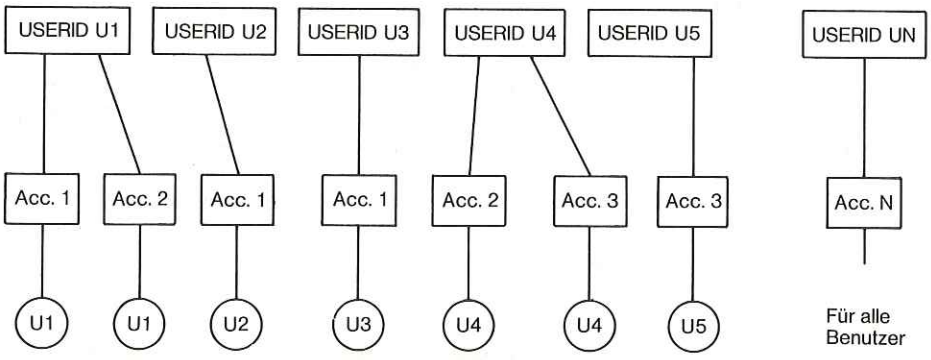



Bild 2

Anzahl der Dateien unter jeder Benutzerkennung U: etwa 60.

Unter einer besonderen Benutzerkennung UN sollten alle Dateien katalogisiert werden, welche sich nicht in das obige Schema einfügen lassen. Das sind Dateien, welche von mehreren der neuen Benutzerkennungen benutzt werden müssen (EXEC-Dateien).

Durch diese Konfiguration wird eine Benutzergruppe mit einem „Hauptbenutzer" unter einer besonderen Benutzerkennung eingerichtet.

Dabei ist zu bemerken, daß die Anzahl von Benutzerkennungen im System **nicht** begrenzt ist und ein Leistungsverlust durch eine größere JOINFILE zu vernachlässigen ist (ISAM-Dateiverarbeitung).

Jedoch ist die Anzahl von Abrechnungsnummern je Benutzerkennung auf 60 begrenzt (Q.F STARTUP-Parameter). Die Abrechnungsnummer wird nur für Abrechnungszwecke benutzt und hat keine Bedeutung für die Systemverarbeitung.

In Tabelle1 wurde das „optimale" RZ gestrichelt angegeben, wobei sicherlich noch Verbesserungen entsprechend diesem Beitrag möglich sind.

| Benutzer RZ | Anzahl der Benutzer-kennungen | Anzahl der Dateien unter TSOS | % von Benutzern weniger als 60 Dateien |
|---|---|---|---|
| C1 | 215 | 1050 | 82 |
| C2 | 64 | 522 | 72 |
| C3 | 51 | 212 | 65 |
| C4 | 361 | 621 | 85 |
| C5 | 21 | 280 | 48 |
| C6 | 103 | 277 | 52 |
| C7 | 319 | 449 | 84 |

Tabelle1: Anzahl der Benutzerkennungen, Anzahl der Dateien unter der Benutzerkennung TSOS.

### 3. Sekundärspeicherzuordnung

Die Sekundärspeicherzuordnung ist sehr zeitaufwendig, da bei jeder Anforderung der entsprechende Katalogeintrag geändert wird.

Dieser Mechanismus beeinträchtigt die Benutzer- und Systemleistung, da der Zugriff über die Katalogverwaltung zur Koordinierung mit längeren Sperrungen verbunden ist.

Die Tabelle 2 zeigt die ermittelten Werte für die Sekundärspeicherzuordnung. Es zeigt sich, daß die meisten Dateien mit dem Systemwert von 3 PAM-Seiten arbeiten. Für die Benutzer C2 und C7 scheint der Systemwert auf 15 PAM-Seiten gesetzt zu sein.

Dies bedeutet, daß die Benutzer sorgfältiger die Werte für die Sekundärspeicherzuordnung auswählen sollten, da sonst mit dem Systemwert gearbeitet wird.

| Benutzer RZ | Anzahl der Dateien mit dem Sekundär-Wert von: 0 | 3 | 6 | 9 | 12 | 15 | 30 | größer 30 |
|---|---|---|---|---|---|---|---|---|
| C1 | 245 | 7819 | 563 | 8 | 34 | 20 | 92 | 121 |
| C2 | 804 | 1084 | 325 | 0 | 1 | 1942 | 47 | 58 |
| C3 | 31 | 1700 | 278 | 0 | 17 | 2 | 15 | 35 |
| C4 | 4 | 7985 | 743 | 3 | 3 | 128 | 31 | 80 |
| C5 | 339 | 1481 | 373 | 0 | 0 | 7 | 1 | 6 |
| C6 | 412 | 4953 | 1033 | 0 | 19 | 16 | 12 | 21 |
| C7 | 722 | 2988 | 716 | 17 | 47 | 4740 | 114 | 426 |

Tabelle2: Werte für Sekundärspeicherzuordnung

Der Systemverwalter hat einige Möglichkeiten, die häufige Anwendung des obigen Mechanismus zu verhindern:

**a) Systemmerkmale:**

Zur SYSGEN-Zeit (PARAM-Karte für DMSCALL) kann der Systemverwalter den Systemwert für die Sekundärspeicherzuordnung modifizieren (siehe Tabelle 2, Benutzer-RZ C2, C7).

Ein Wert von 15 PAM-Seiten ist ein Maximum.

Durch einen höheren Systemwert kann die Anzahl der Sekundärzuordnungen zwar verringert werden, jedoch sollten damit die folgenden Maßnahmen verbunden sein:

● Automatische Freigabe:
Der Systemverwalter sollte mittels einer Prozedur die unbenutzten Speicherseiten der Benutzerdateien freigeben, um einer Sättigung des gemeinschaftlichen Speicherbereichs vorzubeugen. Der Aufbau dieser Prozedur sollte den jeweiligen Anforderungen des Rechenzentrums angepaßt sein.

● Benutzermaßnahme:
Die Benutzer sollten angewiesen werden, nach Dateiaufbau die unbenutzten Seiten wieder freizugeben. Diese Möglichkeit besteht durch die Angabe eines negativen SPACE-Wertes im FILE-Kommando/Makro für eine erstellte Datei.

**b) Benutzerhinweise:**

Der Systemverwalter sollte die Benutzer bei der Speicherbelegung (PRIMÄR-Zuordnung) und bei der Erweiterung von Dateien (SEKUNDÄR-Zuordnung) beraten.

Beispiel: Angenommen, eine Datei ist mit mindestens 100 PAM-Seiten einzurichten, könnte aber auf etwa 200 PAM-Seiten anwachsen.
Der Benutzer muß dann folgendes angeben:

```
/FILE  Dateiname, SPACE=(120,32)
Ablauf des Erstellungsprogrammes
/FILE  Dateiname, SPACE=(-200)
```

Durch die PRIMÄR-Zuordnung von 120 PAM-Seiten wird die SEKUNDÄR-Zuweisung bei der Erstellung vermieden und auch nicht zu viel gemeinschaftlicher Speicherplatz belegt. Durch die SEKUNDÄR-Zuweisung von 32 PAM-Seiten werden höchstens drei Zuweisungen vorgenommen. Am Ende des Auftrags werden die zusätzlich belegten Speicherseiten wieder freigegeben.

**4. Größe der Datei TSOSCAT**

Die maximale Größe der Datei TSOSCAT beträgt 4096 PAM-Seiten. Davon benutzt die Katalogverwaltung selbst die ersten freien Seiten.

Die Größe des Katalogs sollte der Systemverwalter sorgfältig planen. Diese kann mit den folgenden Angaben ermittelt werden:

● Anzahl der Benutzerkennungen
● Anzahl der Dateien je Benutzerkennung

Geht man davon aus, daß je Katalog-PAM-Seite etwa 15 Katalogeinträge enthalten sind, so kann man z. B. folgende Berechnung anstellen:

Größe des Katalogs

$= \frac{60}{15} \cdot$ Anzahl Benutzer-ID. Hauptzuordnung für 60 Dateien

$+ \frac{30}{15} \cdot$ Anzahl Benutzer-ID. Vorgabe für 30 weitere Dateien

$+ 60$ Sicherheitsfaktor für alle Benutzerkennungen

Die ermittelten Werte der Benutzerkataloge waren etwas seltsam und sind in Tabelle 3 aufgezeigt.

| Benutzer-RZ | Anzahl der tatsächlich verwend. Seiten | Größe von TSOSCAT | Berech. Werte | Differ. |
|---|---|---|---|---|
| C1 | 680 | 6000(?) | 1350 | 4650 |
| C2 | 369 | 1041 | 444 | 597 |
| C3 | 196 | 759 | 366 | 393 |
| C4 | 850 | 1323 | 2226 | – |
| C5 | 162 | 720 | 186 | 534 |
| C6 | 630 | 720 | 678 | 42 |
| C7 | 866 | 5922(?) | 1974 | 3948 |

Tabelle 3: Größe der Datei TSOSCAT

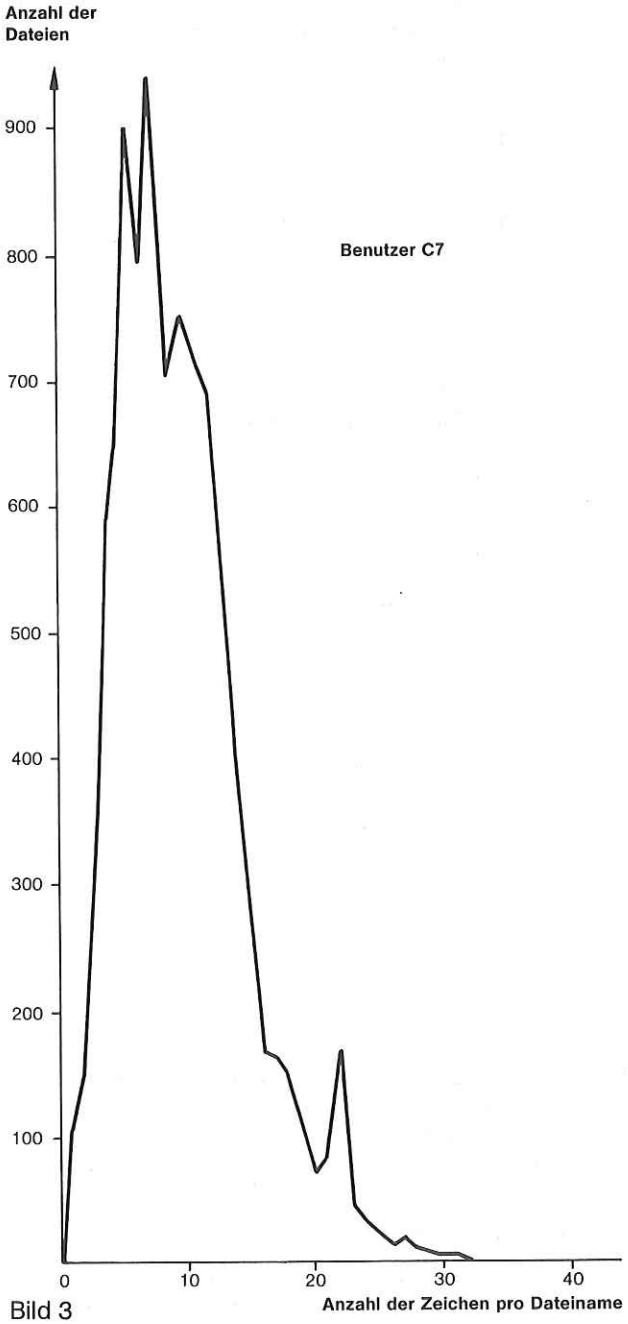



Bild 3

Die Größe der Kataloge der Benutzer C1 und C7 übersteigt die Größe, die die BS2000-Katalogverwaltung unterstützen kann!! Die Speicherzuordnung über 4096 PAM-Seiten hinaus ist nutzlos für die Katalogverarbeitung und belastet nur den gemeinschaftlichen Speicherbereich (public space.)

### 5. Länge der Dateinamen

Auch die Länge der Dateinamen hat einen gewissen Einfluß auf die Leistungsfähigkeit des Katalogsystems. Denn je länger die Dateinamen sind, um so geringer ist die Anzahl der Katalogeinträge in einer Katalog-PAM-Seite. Demzufolge ist bei gleicher Anzahl von Dateien eine höhere Anzahl von Ein-Ausgabe-Operationen erforderlich, um einen bestimmten Katalogeintrag zu finden.

Der Systemverwalter und auch jeder Benutzer sollte deshalb möglichst kurze Dateinamen verwenden.

Bild 3 zeigt die Verteilung der Dateinamenlänge des Benutzers C7, die anderen Benutzer haben etwa die gleiche Verteilung.

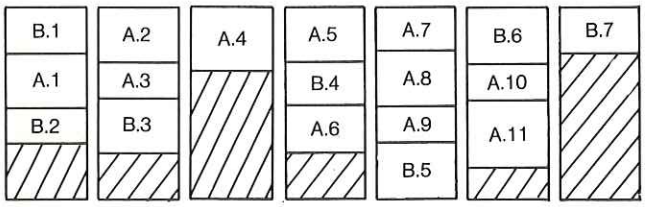
Die meisten Dateien haben kurze oder mittlere Dateinamenlänge.

### 6. Löschen von Dateien unter einer Benutzerkennung

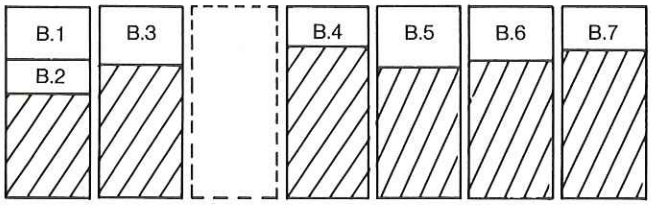
Am Ende einer Arbeitsphase oder eines Projektes tritt oft der Fall auf, daß einige Dateien gelöscht werden und damit „Löcher" in den einer Benutzerkennung zugeordneten Katalogseiten entstehen. Solange die Seite nicht vollständig leer ist, wird sie auch nicht an den Pool der freien Katalogseiten zurückgegeben. Damit wird eine relativ hohe Anzahl von Ein-Ausgabe-Operationen benötigt, um einen bestimmten Eintrag zu finden.
Die Katalogverwaltung füllt zwar die „Löcher" bei der Erstellung von neuen Dateien wieder auf, jedoch kann dies geraume Zeit dauern. Diese Situation kann durch Auslagern aller Dateien auf Band (mittels TSOSMT/SYSUPD) verbessert werden. Danach werden die Plattendateien gelöscht und wieder von Band auf Platte zurückgespeichert.

Dieser Vorgang wird im nächsten Beispiel dargestellt:

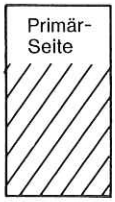
**Zustand 1.** Der Benutzer hat 7 Katalogseiten für seine Dateien A. und B. belegt.



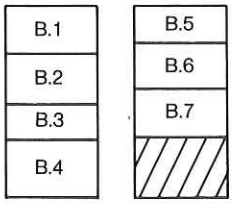

**Zustand 2.** Der Benutzer hat alle Dateien A. gelöscht, belegt aber immer noch 6 Katalog-PAM-Seiten.




**Zustand 3.** Der Benutzer stellt die restlichen Dateien auf Band sicher und löscht die Plattenkopien.




**Zustand 4.** Der Benutzer speichert alle Dateien von Band auf Platte zurück.




Damit werden nur noch zwei Katalogseiten benötigt.

## 7. Katalogschutz

Der Katalog TSOSCAT ist gegen den Zugriff von Benutzern mit einem Lese- und einem Schreibpaßwort geschützt. Falls ein Benutzer die Paßworte kennt, erhält er Zugriff zu allen privaten Informationen, wie z. B. weiteren Paßwörtern anderer Dateien unter anderen Benutzerkennungen und kann den Katalog-inhalt verändern.

Deshalb sollten die Paßwörter mit dem /CATAL-Kommando vom Systemverwalter öfter geändert werden. Bei Bedarf werden Paßwörter dem Systemverwalter mit /FSTAT, P-Kommando ausgegeben.

Das BS2000-System selbst benötigt keine Paßworte beim Zugriff auf den Systemkatalog.

Bild 4 zeigt den Zugriff zum Katalog durch das System, einen normalen Benutzer und den TSOS-Benutzer.

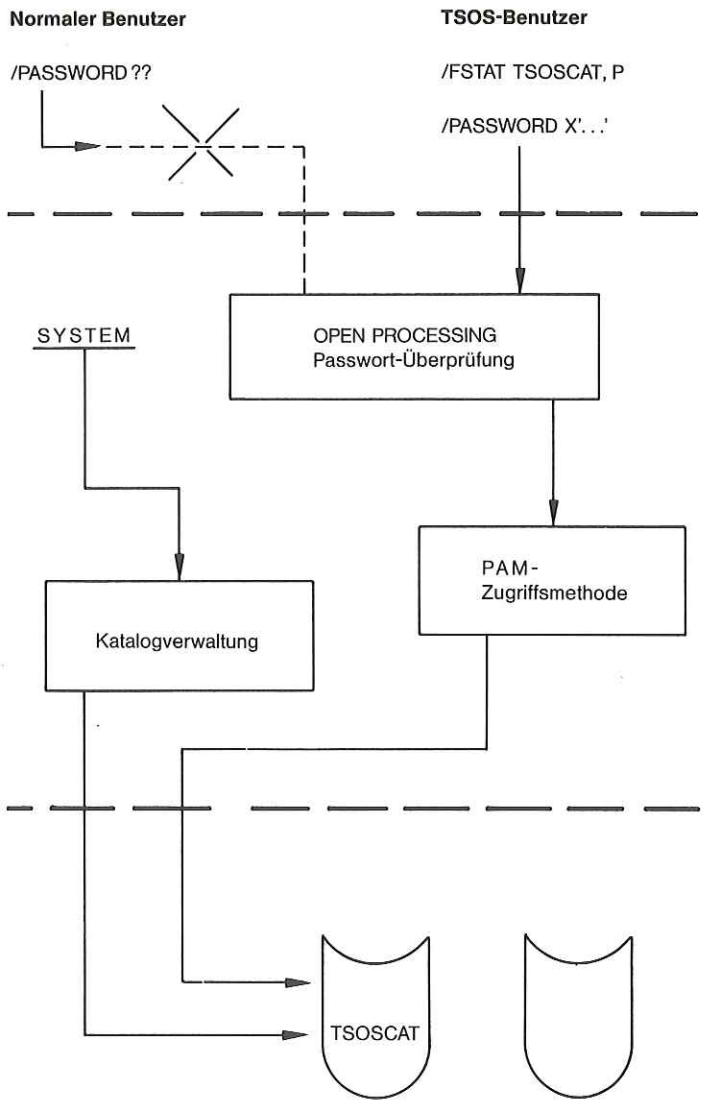



Bild 4

## Schlußwort

### Systemverbesserungen

In den nächsten Versionen des BS2000 wird eine Durchsatzverbesserung für Katalogzugriffe realisiert.

Durch eine Erweiterung soll die hohe Anzahl von E/A-Operationen auf den Katalog und damit auf den gemeinschaftlichen Speicherbereich begrenzt werden.

Dies wird durch einen neuen Katalogmakro mit direktem Zugriff zu einem bestimmten Eintrag erreicht. Damit werden eine Reihe von Suchoperationen im Katalog vermieden, welche vor allem bei Sekundärspeicherzuordnung und beim Schließen einer Datei auftreten.

### Literaturhinweise

**BS2000 V3.0**
Datenverwaltungssystem (DVS)
Beschreibung

**BS2000 V3.0**
Systemgenerierung (SYSGEN)
Beschreibung

**BS2000 V3.0**
Kommandosprache des Organisationsprogramms
Beschreibung

# Kurzreferat für die Dokumentation

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Jelske Kloppenburg

## Dialogprogramme in ALGOL

Es wird erläutert und mit Beispielen belegt, wie man beim Programmieren von Dialogprogrammen in ALGOL60 einen 'Vorschub' bewirkt, damit der Text sofort auf das Datensichtgerät ausgegeben wird.

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Walter Holzer

## RPG2 Systemergänzungen für BS1000

Für die Anwendung in BS1000 RPG2-Programmen werden sechs Unterroutinen vorgestellt und beschrieben. Sie ermöglichen im einzelnen z. B. Abruf von abgelaufener CPU- und aktueller Tageszeit, Identifikation von Betriebssystem, ZE, Programmname, Aktivierung des PDUMP-Makros, Simulation von Ein-Ausgabedateien, Lesen von Daten oder Parametern aus dem MONIT2-Bereich und Ein-Ausgabe über Konsole.

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Helena Geißler

## Checkpoint-Restart-Routine (CKPTIN)

Das Programm CKPTIN ermöglicht es dem Operateur, jeden beliebigen Batch-Job mit /INTR abzubrechen und durch /RESTART zu beliebiger Zeit wieder zu aktivieren, wie das häufig nötig ist. Die Verwendung von Band-Dateien ist nur bedingt, der Einsatz für EAM-Files nicht möglich.

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Werner Herrmann

## Datenschutz im BS2000

Der Beitrag beschreibt und erklärt Maßnahmen, die im BS2000 den Schutz der bis zu 119 möglichen Benutzer und ihrer Dateien sowie der Programme untereinander gewährleisten. Der Systemverwalter ist hierbei ein privilegierter Benutzer mit erweiterten Möglichkeiten.

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Heinz Kröger

## Der Klassentest in ANS-COBOL

Probleme beim Klassentest, die sich aus Mängeln im Lochkartencode, durch Anordnung der Lochertastaturen sowie durch das Eingabesystem ergeben, werden erläutert und es werden entsprechende Maßnahmen zur Abhilfe anhand von Beispielen beschrieben.

**data spezial** **SIEMENS**

Systeminformationen, Ausgabe 1/1977

Michel Defamie

## Maßnahmen zur Leistungsverbesserung der Katalog-Verwaltung im BS 2000

Es werden Maßnahmen beschrieben, durch die Systemverwalter und Benutzer des BS2000 die Leistungsfähigkeit der Katalog-Verwaltung verbessern können. Dazu wurden die Kataloge von sieben BS2000-Benutzern kritisch untersucht, anhand ihrer wesentlichen Bereiche beurteilt und zu Verbesserungsvorschlägen herangezogen.