CPG2

Programmiererhandbuch

I n h a l t s v e r z e i c h n i s Seite

30 CPG-Information 0030

Abschnitt 1 Allgemeines 1200

1200 Die Sprache CPG 1200
1800 Voraussetzungen 1800
1810 Programmierkenntnisse 1810
1820 Software-Voraussetzungen 1820

Abschnitt 2 Service Funktionen 2000

2100 Datenfelder 2100
2110 Feldnamen 2110
2120 Feldgruppen 2120
2130 CPG-interne Felder 2130
2140 Speicherungsarten 2140
2150 Bildschirm-Eingabefelder 2150
2160 Zwischenspeichern von Daten 2160
2190 Zwischenspeichern auf Temporary Storage 2190

2200 Blockschaltbild 2200
2260 Schalter 2260
2270 Gruppenstufen 2270

2300 Dateiverarbeitung 2300
2330 DL/I-Schnittstelle 2330
2340 Datenviews 2340
2350 DC-Funktionen 2350

2400 Programmierhilfen 2400
2410 Entscheidungstabellen 2410
2420 Feldaufbereitung 2420
2450 Strukturierte Programmierung 2450
2470 Datenstrukturen 2470

2500 Optimierung der TWA-Größe 2500
2550 Transaktionsorientierte Programmierung 2550

2600 Data Dictionary im Programm 2600

2800 Testhilfen 2800
2810 DEBUG 2810
2830 SDUMP 2830

2900 Einschränkungen 2900
2970 Vorbereitungen auf den ESA-Mode 2970

Abschnitt 3 Programmentwurf 3000

3010 Syntax 3010
3020 CPG2 und starre Schreibweise gemischt 3020
3030 Gliederung 3030
3300 Options Division 3300
3400 Files Division 3400
3500 Data Division 3500
3600 Forms Division 3600
3700 Input Division 3700
3800 Procedure Division 3800
3810 Grammatik der Procedure Division 3810
3900 Output Division 3900

Abschnitt 4 Operationen 4000

Abschnitt 5 HL1-Programmierung 5000

Abschnitt 6 Ausführung 6000

6300 Batch-Programmierung 6300

6900 Fehlermeldungen während der Umwandlung 6900
6930 Fehlermeldungen bei der Assemblierung 6930
6950 Fehlermeldungen während der Ausführung 6950
6980 Batch-Fehlermeldungen

Abschnitt 7 Tabellen, Begriffe, Zubehör 7000

7000 Übersicht der Rechenbestimmungen 7000
7020 Bildschirmattribute 7020
7030 Write Control Character (Bildschirm) 7030
7040 Übersicht der Maximalwerte 7040
7050 Zwischenspeicher (CWA, TCT, Temp. Stor.) 7050
7070 Aufbereitungsschlüssel (Edit Codes) 7070

7300 Regeln der Grammatik 7300
7400 Schlüsselworte der Procedure Division 7400

7500 Zubehör 7500

Abschnitt 8 Beispiele 8000

8000 Platten-Änderung 8000
8010 Dateianzeige mit READ-PAGE 8010
8015 Dateiänderung mit UPDAT und WRITE 8015
8017 Dateiänderung mit EXCPT 8017
8020 RRDS-Datei mit numerischem Keyfeld 8020
8025 ESDS-Datei 8025
8030 Hinzufügen zu einer ESDS-Datei 8030
8035 Drucken im Line Mode / Buffer Mode 8035
8040 Feldaufbereitung mit EDIT 8040
8045 READ-BACK 8045
8050 Update bei variabler Satzlänge 8050
8060 Temporary Storage Queuing 8060
8065 variable Cursorpositionierung 8065
8070 Operation FIND 8070

	8075	Variabler Mapname	8075
	8080	TWA-LOAD und TWA-SAVE	8080
	8100	Dateipflege, dialogorientiert	8100
	8110	Dateipflege, taskorientiert	8110
	8120	CONVERT	8120
	8125	Konvertieren von Zeit- und Datumsfeldern	8125
	8130	Datenaustausch über CPGCOM	8130
	8140	Segmente bei sequentieller Verarbeitung	8140
	8145	Segmente bei direkter Verarbeitung	8145
	8150	Logisch verknüpfte IF/DO (Syntax)	8150
	8160	Groß-/Kleinschreibung taskorientiert	8160
	8190	Massen-Insert im CICS für VSAM-Dateien	8190
Batch-Beispiele			8600
	8600	Leser, Drucker, Überlaufsteuerung	8600
	8602	Programmkatalog CPGWRK	8602
	8603	Vom Kartenleser nach KSDS laden	8603
	8604	Kopieren von KSDS nach KSDS	8604
	8605	Direktes Update in einer KSDS-Datei	8605
	8606	Sequentielles Update in einer KSDS-Datei	8606
	8607	Sequentielles Löschen (KSDS)	8607
	8608	Direktes Löschen (KSDS)	8608
	8609	Programmkatalog rückwärts auflisten	8609
	8610	Kopieren vom Kartenleser nach ESDS	8610
	8611	Drucken einer ESDS-Datei	8611
	8612	Sequentielles Update (ESDS)	8612
	8613	ESDS direkt und sequentiell verarbeiten	8613
	8614	Drucken einer RRDS-Datei	8614
	8615	RRDS-Datei direkt verarbeiten	8615
	8617	Laden RRDS vom Kartenleser	8617
	8618	Seuentielles Update einer RRDS-Datei	8618
	8619	Datei mit OPEN im Programm	8619
	8621	Ausgabe auf Stanzer	8621
	8622	Kopieren von Platte auf Band	8622
Abschnitt 9	9900	Stichwortverzeichnis	9900

Information 0030

Dieses Handbuch wird herausgegeben als Programmierer-Handbuch für das Anwendungsentwicklungssystem CPG.

Anschrift: Lattwein GmbH
Otto-Brenner-Straße 25
52353 Düren

Telefon: 02421 81051

Telefax: 02421 82127

Internet: http://www.lattwein.de

E-Mail: service@lattwein.de

Auskunft: In der unten angegebenen Arbeitszeit steht den Benutzern des CPG ein zentraler Telefon-Wartungsdienst zur Verfügung, der bemüht ist, alle Fragen zu beantworten, die in diesem Handbuch nicht behandelt wurden.

Arbeitszeit: 8:30 – 16:00 Uhr

Abschnitt 1 Allgemeines 1200

Die Sprache CPG 1200

Die Sprache CPG ist flexibel in ihrer Syntax und passt sich weitgehend den Vorkenntnissen des Programmierers an. Neben dem CPG-Format, das im wesentlichen der Schreibweise der heute gängigen Programmiersprachen entspricht, ist auch eine spaltengebundene Schreibweise unterstützt, die RPG-kompatibel ist.

Die Aufgabe: Wenn X gleich Y, dann rechne A * B, stelle das Ergebnis in C ab und lösche B kann in CPG wie folgt beschrieben werden:

CPG-Schreibweise:

```
IF X = Y
   C = A * B
   B = 0
END
```

RPG III Schreibweise:

```
C          X      IF   Y
C          A      MULT B       C
C                 Z-ADD0       B
C                 END
```

Voraussetzungen 1800

1.Programmier-Kenntnisse 1810

Voraussetzung für die Programmierung mit CPG2 ist die Beherrschung einer höheren Programmiersprache, da die Grundzüge des Programmierens in dieser Dokumentation als bekannt vorausgesetzt werden.

Darüber hinaus sind TP-Kenntnisse oder -Erfahrungen von Vorteil, jedoch nicht Bedingung.

2.Software-Voraussetzungen 1820

CPG erzeugt ein Assembler-Quellen-Programm. Vorausetzung für die Umwandlung ist daher die Verfügbarkeit eines Assembler Compilers.

Das generierte Assembler-Quellenprogramm kommt dabei mit dem Instruk-

tionssatz der IBM 370 aus. Es kann auf allen Anlagen der /370- und der /390-Architektur sowie auf gleichwertigen Maschinen anderer Hersteller eingesetzt werden, soweit für diese ein IBM kompatibler Assembler Compiler zur Verfügung steht.

Für die Standard-Version ist ein beliebiges TP-Steuerprogramm erforderlich, dessen Ein- und Ausgabe-Makros vom Benutzer oder Hersteller in einem Interface zusammengefasst werden. Für CICS wird dieses Interface mit CPG mitgeliefert.

Abschnitt 2 Service-Funktionen 2000

Datenfelder 2100

Feldnamen und Felddefinitionen 2110

Jedem Feld in einem CPG2-Programm ist ein Bereich in der Transaction Work Area (TWA) zugeordnet. Ein Feld kann explizit oder implizit definiert werden.

Die explizite Definition von Feldern erfolgt in der Data Division.

Wird in der Input Division beschrieben, aus welchen Positionen eines Datenträgers ein Feld gelesen werden soll, so erfolgt dadurch gleichzeitig eine implizite Felddefinition

Alle in der TWA abgestellten Felder sind entweder alphanumerisch oder gepackt numerisch. Immer wenn ein Feld definiert wird, verlangt CPG2 die folgenden Angaben:

a) Feldname
b) Feldlänge
c) Format (numerisch oder alphanumerisch)
d) Falls numerisch: die Anzahl Dezimalstellen.

Feldnamen können bis zu 30 Stellen lang sein. Sie dürfen keine Leerstellen (Blanks) enhalten. Das erste Zeichen muss ein Buchstabe von A bis Z sein. Es wird empfohlen, Feldnamen mit mehr als 6 Stellen nur in Ausnahmefällen zu verwenden, da der Compiler bei mehr als 6 Stellen einen internen Namen generiert. Somit wird bei Verwendung langer Feldnamen der Einsatz von Data Dictionary-Strukturen und programmexternen QSF-Masken erschwert. Diese Werkzeuge lassen zur Zeit nur 6-stellige Feldnamen zu.

Das $-(Dollar-)Zeichen ist an beliebiger Stelle im Feldnamen erlaubt.

Die Zeichenfolge CPG auf den ersten drei Stellen eines Feldnamens ist reserviert. Die Benutzung ist nur mit den im Abschnitt 2130 angegebenen CPG-internen Feldern zugelassen.

Bei der Feldgruppenverarbeitung ist zu beachten, dass auch für die Darstellung von Feldgruppenelementen eine Beschränkung gilt: Die Bezeichnung setzt sich aus dem Namen der Feldgruppe und dem Index in Klammern zusammen und darf maximal 7 Stellen lang sein.

Die maximale Länge eines alphanumerischen Feldes beträgt 256 Bytes. Die maximale Länge eines numerischen Feldes beträgt 8 Bytes gepackt, d.h. 15 Ziffern.

Alle Felder sind standardmäßig alphanumerisch, wenn nichts anderes angegeben wird. Felder werden als numerisch gekennzeichnet, indem angegeben wird, dass sie 0 oder mehr Dezimalstellen besitzen.

- OTTO 5. bedeutet, das Feld OTTO ist alphanumerisch und 5 Stellen lang.
- HUGO 3 0. bedeutet, das Feld HUGO ist numerisch, 3 Stellen lang und hat Null Dezimalstellen.

Durch Datenstrukturen und Überlagerungen in der TWA können Felder und Feldgruppen unter anderem Namen zu Gruppen oder Sätzen zusammengefasst werden.

Wird ein Feld in der Input Division spezifiziert, ohne dass es zuvor in der Data Division definiert wurde, so gilt es als implizit definiert. CPG errechnet die Länge des Feldes aufgrund der angegebenen ersten und letzten Stelle und ordnet entsprechend Speicherplatz in der TWA zu. Ist eine bestimmte Anzahl Dezimalstellen für das Feld spezifiziert (d.h. 0 oder mehr), so behandelt CPG es als gepacktes Feld.

Werden die Eingabedaten für ein Feld als binär oder im ungepackten Format gekennzeichnet, so wandelt CPG sie automatisch in das gepackte Format um, bevor es sie in dem entsprechenden Feld in der TWA abspeichert.

Bei Datenstrukturen sind numerische Felder nur in gepacktem Format zugelassen.

Ist im CPG-Programm zuvor ein Feld mit demselben Namen explizit oder implizit mit anderer Feldlänge definiert worden, so wird ein Syntaxfehler erkannt und während der CPG-Übersetzungsphase kommentiert. Sollte das Feld numerisch sein, so muss für die Anzahl Dezimalstellen ein Wert von 0 oder größer eingetragen werden. Sind die für das Feld definierten Attribute nicht kompatibel mit der Operation, bei der es benutzt wird, so wird ebenfalls ein Fehler angezeigt.

Folgenden Hinweise zu Felddefinitionen müssen beachtet werden:

a) Alle Felder müssen irgendwo in dem Programm, in dem sie benutzt werden, explizit oder implizit definiert werden. (Mit Ausnahme der CPG-internen Felder und UDATE und UTIME).

b) Ein Feld kann nicht in einer Weise benutzt werden, die seinen definierten Attributen widerspricht.

Regeln für Feldnamen: ----------

Feldnamen dürfen maximal 30 Stellen lang sein.

Das erste Zeichen eines Feldnamens muss ein Alphazeichen oder Dollar sein.

Ab der zweiten Stelle dürfen Feldnamen beliebige Sonderzeichen enthalten.

Feldnamen, die Sonderzeichen (außer Dollar) enthalten und/oder die länger als 6 Stellen sind, werden in sechsstellige Namen (CPxxxx) umgesetzt.

Beispiele:

DIES-IST-EIN-GÜLTIGER-NAME	richtig
DIES-IST-KEIN-GÜLTIGER-FELDNAME	falsch (zu lang)

A*27+35=89:IST-NICHT-GUT:ABER	richtig
7-MEILEN-STIEFEL	falsch (Stelle 1 nicht Alpha)
$$10	richtig
A-100	richtig

Feldgruppen 2120

Feldgruppen sind Gruppen von Feldern gleicher Länge und gleichen Typs, die beispielsweise über eine DO-Schleife indiziert verarbeitet werden sollen.

Definition

Die Definition einer Feldgruppe erfolgt ausschließlich in der Data-Division in der Form: Name, Anzahl Elemente, *, Länge, Dezimalstellen

Alphanumerische Feldgruppen werden mit Blanks, numerische mit Nullen vorformatisiert.

In einem CPG2-Programm sind maximal 80 Feldgruppen unterstützt.

Eingabe / Ausgabe

Es können ganze Feldgruppen oder Einzelfelder mit festem Index eingelesen und ausgegeben werden. Variable Indizes sind in Input-Division und Output-Division nicht unterstützt.

1. Bildschirm

Feldgruppen werden in den Ausgabebestimmungen wie Einzelfelder behandelt. Beim Ausgeben auf Bildschirm-Dateien wird die letzte Stelle des ersten Feldgruppen-Elements angegeben. Die folgenden Elemente werden automatisch in den folgenden Bildschirm-Zeilen angezeigt. Das Einlesen vom Bildschirm geschieht analog.

2. Platten-Dateien / Druckausgabe / Feldaufbereitung mit EDIT - SELCT

Für andere Dateien wird immer für die Ausgabe die letzte Stelle des letzten Feldguppenelements angegeben, für das Einlesen der Bereich vom ersten bis zum letzten Byte der Feldgruppe.

Werden ganze Feldgruppen mit einem Aufbereitungsschlüssel ausgegeben, so sind die Einzelfelder zu der erforderlichen Stellenzahl für die Aufbereitung noch zusätzlich durch 2 Blanks getrennt.

Regeln für Feldgruppen-Namen: ----------

Für indiziert verarbeitete Feldgruppen gelten folgende Regeln:
1. Länge Feldgruppenname + Länge Indexname muss kleiner als 6 sein.
Beispiele:

A(I)	richtig
A(ZAHL)	richtig
FG(88)	richtig

oder:

2. Der Indexname ist einstellig, der Feldgruppenname bis zu 27 Stellen lang (es wird ein CP-Name für die Längen 7 bis 27 generiert bzw. für kürzere Feldgruppennamen, die Sonderzeichen enthalten).

Beispiele:

FELDGRUPPE(A)	richtig
FG-2(K)	richtig
FGRUPPE(I2)	falsch, Index nicht einstellig

Verarbeitung

Folgende Operationen der Rechenbestimmungen sind indizierbar:

alle numerischen Operationen

CAB	Compare and Branch
CAS	Compare and Branch Subroutine
CBS	Compare and Branch Subroutine
COMP	Compare
CONVT	Konvertieren eines Alphafeldes
DELC	Zeichen entfernen
DO	Schleifen
EDIT	Feld aufbereiten
ELIM	Zeichen löschen und durch Blank ersetzen
FILL	Alphafeld füllen
IF	Wenn - Abfrage
MOVE	Rechtsbündig übertragen
MOVEA	Feldgruppe übertragen
MOVEL	Linksbündig übertragen
MOVEN	Alphafeld in numerisches Feld übertragen
REPLC	Blank durch Zeichen ersetzen
SCAN	Suchen einer Zeichenkette
SELCT	Feld auswählen

D.h. in einem gültigen Operanden einer dieser Operationen kann anstatt eines Feldnamens der Name einer Feldgruppe angegeben werden. Dabei gelten folgende Regeln:

1. Feldgruppe mit festem Index

wurde eine Feldgruppe mit dem Namen 'FGR' definiert, so bewirkt eine Operation:

- FGR(7) = X

dass der Wert 'X' in das 7. Feld der Feldgruppe 'FGR' addiert wird. Dabei ist zu beachten, dass die Gesamtlänge der Eintragung nicht größer als 7 Stellen sein darf. Der Name der Feldgruppe darf also in diesem Fall maximal 4 Stellen groß sein.

2. Feldgruppe mit variablem Index

Wurde eine Feldgruppe mit dem Namen FGR definiert, so bewirkt eine Operation:

- FGR(N) = X

dass der Wert X in das N-te Feld der Feldgruppe FGR übertragen wird, wobei das Feld mit dem Namen 'N' eine ganze Zahl zwischen 1 und der Anzahl der für die Feldgruppe definierten Felder enthalten muss. Jeder andere Wert führt zum abnormalen Abbruch des Programms. Die Gesamtlänge der Eintragung darf nicht größer als 7 Stellen sein.

3. Feldgruppe ohne Index

Wurde eine Feldgruppe mit dem Namen FGR definiert, so bewirkt die Operation:

- FGR = 0

dass alle Elemente dieser Feldgruppe auf 0 gelöscht werden. Wurde eine zweite Feldgruppe mit dem Namen FG2 definiert, so bewirkt die Operation:

- FG2 = FGR

dass alle Elemente der Feldgruppe FGR in die entsprechenden Elemente der Feldgruppe FG2 übertragen werden, wenn die Anzahl der Elemente für beide Feldgruppen gleich groß ist. Ist eine der beiden Feldgruppen kleiner als die andere, so werden nur so viele Elemente übertragen, wie für die kleinere Feldguppe definiert wurden.

Achtung:

In der beschriebenen Form können nur numerische Werte zwischen numerisch definierten Feldern und Feldgruppen übertragen werden.

Zur Übertragung alphanumerischer Werte stehen die verschiedenen MOVE-Operationen zur Verfügung.

CPG-interne Felder 2130

CPG-interne Felder sind bereits vom Compiler vorgegeben und dürfen daher vom Programmierer nicht definiert werden. Folgende Felder stehen zur Verfügung:

CPGATR Variables Attribut (wenn QSF nicht genutzt wird)

Definition bei Bildschirmausgaben mit der Eintragung 'VAR'. In diesem einstelligen Feld wird das gewünschte Attribut eingetragen.

CPGBZL Bezugszahlenleiste

CPGBZL ist ein 100-stelliges Feld, das den Zustand der Bezugszahlen von 00 bis 99 enthält. Es kann beispielsweise dazu benutzt werden, die Bezugszahlen vor einem Unterprogrammaufruf zu retten.

CPGCOM Common Area.

Dieser Bereich kann über EDIT- und SELCT-Operationen bei Programmverbindungen mit Command Level als Zwischenspeicher verwendet werden.

CPGCSA Common System Area.

Dieser Bereich kann über EDIT- und SELCT-Operationen verändert oder abgefragt werden. Die in der CPGCSA stehenden Daten stehen unabhängig vom Programm oder Terminal allen Systembenutzern zur Verfügung. Die Länge des Feldes entspricht der Länge der im TP-Steuerprogramm generierten Common Work Area. (Anhang der Common System Area). Die maximale Länge beträgt 3584 Stellen.

CPGCUR Variable Cursor-Position (wenn QSF nicht genutzt wird)

Definition bei Bildschirmausgaben mit Eintragung 'VAR'. In diesem 4 -stelligen Feld wird die gewünschte Cursor-Position eingetragen.

CPGDID Variabler Druckername

In diesem 4-stelligen Feld wird der Name des Druckers eingetragen, wenn in der Files Division ein variabler Drucker vereinbart wurde.

CPGDRC DL/I Return-Code.

Dieser Bereich enthält den DL/I-Return-Code nach einem DL/I-Call.

CPGEOJ End of Job. Dieses vierstellige numerische Feld ermöglicht es, aus Batchprogrammen einen variablen Returncode an VSE oder z/OS zu übergeben, der im Conditional Job Control abgefragt werden kann.

CPGFIS Verschiebefaktor bei Dateieingaben.

Dieses fünfstellige numerische Feld enthält den Ver-

schiebefaktor, falls bei der Eingabe-Satzbestimmung vereinbart wurde, dass die Eingabepositionen variabel angewandt werden können. Der Inhalt des Feldes CPGFIS wird in diesem Fall auf die angegebenen Eingabepositionen addiert. (Siehe Beispiel in Kapitel 8000)

CPGFRC File Return-Code nach Datei-Operationen

Bei der Programmierung von Datei-Operationen kann auf Schalter ganz verzichtet werden. Nach den Operationen enthält CPGFRC die folgenden Operationen, die anstelle von Schaltern abgefragt werden können:

DK	Duplicate Key nach READ auf eine AIX-Datei
DR	Duplicate Record nach dem Hinzufügen
EF	End of File nach READ, READ-BACK und SETLL
NC	Not Closed nach CLOSE
NF	Not Found nach CHAIN oder CHECK, OPEN, CLOSE
NO	Not Open nach OPEN oder CHECK
OD	**Open Disabled nach CHECK**
Blank	wenn keine Besonderheit festgestellt wurde.

CPGIFC Interface Communication Area

Dieses 32-stellige Feld kann Informationen für das Steuerprogramm-Interface der Methodenbank aufnehmen. CPGIFC wird mit den Operationen EDIT und SELECT verarbeitet.

In Verbindung mit der Operation IFC kann der Programmierer Operationen im Steuerprogramm-Interface selbst modifizieren.

CPGMCI Feldgruppenindex für Cursorposition (im QSF)

Dieses dreistellige numerische Feld enthält den Index der Feldgruppe, die im Feld CPGMCU beschrieben wurde.

CPGMCU Cursor-Position (im QSF)

Dieses 6-stellige Alphafeld enthält den Namen des Feldes, in das der Cursor gesetzt werden soll. Ist dieses Feld leer, wird der Cursor auf das in QSF definierte Feld gesetzt.

CPGMFI Feldgruppenindex bei Lichtstiftauswahl/Bildschirmeingabe

Dieses dreistellige numerische Feld enthält den Index eines Feldgruppenelementes, das mit Lichtstift ausgewählt wurde oder in dem bei der Bildschirmeingabe der Cursor stand (bei Einsatz von QSF).

CPGMFN 1. Feldname bei Lichtstiftauswahl/Bildschirmeingabe

Dieses Feld enthält nach Lichtstiftauswahl den Namen des ausgewählten Feldes bzw. den Namen des Feldes, in dem bei der Bildschirmeingabe der Cursor stand (bei Einsatz von QSF).

2. Begrenzen der Mapausgabe auf bestimmte Zeilen

Dazu wird '$QV' in dieses Feld gesetzt und CPGMLC (wie unten beschrieben) gefüllt.

CPGMLC 1. Feldposition bei Lichtstiftauswahl/Bildschirmeingabe

Dieses vierstellige Alphafeld enthält in den beiden ersten Stellen die Zeile und in den Stellen 3-4 die Spalte des ausgewählten Feldes (bei Einsatz von QSF).

2. Von-Zeile/Bis-Zeile bei Ausgabe von Teil-Maps

Um die Mapausgabe auf bestimmte Zeilen zu begrenzen, bringt man vor der Ausgabe Von-Zeile und Bis-Zeile in das Feld. (CPGMLC = '0205': Ausgabe Zeile 2 bis 5). Das Feld CPGMFN muss gleichzeitig mit '$QV' gefüllt sein.

CPGMPF Dieses zweistellige Feld enthält das Kürzel der zuletzt betätigten Programmfunktionstaste.

CPGMRC Dieses zweistellige alphanumerische Feld hat nach einer Bilschirmeingabe folgenden Inhalt: Entweder ' ' für korrekten Inhalt, 'NI' für NO INPUT oder 'IC' für 'invalid character'. IC wird dann gesetzt, wenn bei der Übertragung vom Bildschirm in ein numerische definiertes Feld ungültige Zeichen oder ungültige Vor- bzw. Nachkommastellen erkannt werden.

CPGNLS National Language Support.

CPGNLS enthält ein D bei deutschen Installationen oder ein E bei englischen (laut Kundenkonfiguration).

CPGPGM Phasenname aus den Options, achtstellig alphanumerisch.

CPGPIW Inhalt der IFC

CPGSIN Systeminformationen (siehe Befehl COMRG)

Mit dem Befehl COMRG CPGSIN können erweiterte Systeminformationen dem Programm zur Verfügung gestellt werden.

CPGSMN Segment-Name und Key-Feedback-Area.

Dieser Bereich enthält die Key-Feedback-Area nach einem DL/I-Call.

CPGTCA Task Control Area

Über dieses Feld kann die TWA des Programms mit EDIT und SELCT verarbeitet werden. Von Stelle 1 bis 100 steht die Bezugszahlenleiste, ab Stelle 117 beginnen die User-Felder. CPGTCA ist maximal 4 K groß.

CPGTCT Terminal Control Table.

Dieser Bereich kann über EDIT- und SELCT-Operationen verändert oder abgefragt werden. Die in der CPGTCT stehenden Daten stehen unabhängig vom verwendeten Programm dem Benutzer eines bestimmten Terminals für die gesamte Laufzeit zur Verfügung. Die Länge des Feldes entspricht der bei der Generierung der Terminal-Control-Table definierten 'User-Area-Length'. Die maximale Länge beträgt 255 Stellen, von denen CPG intern die ersten 10 Stellen benutzt. Im CPG2-Programm sind also maximal

245 Stellen der TCT benutzbar.

CPGTDI Variabler Transient Data-Name

In diesem 4-stelligen Feld wird der Name der Destination eingetragen, wenn in der Files Division variable Transient Data-Verarbeitung vereinbart wurde.

CPGTID Terminal Identification.

In dieses vierstellige Feld wird vom CPG der Name des Terminals aus der Terminal-Control-Table für Abfragen im Programm zur Verfügung gestellt.

CPGTIO Terminal-I/O-Area.

Dieser Bereich kann über eine 'SELCT'-Operation unmittelbar nach Aufruf eines Programms abgefragt werden. Er dient dazu, Daten zusammen mit der Transaction-Identification einzulesen. Die Trans-Id steht dabei in Position 1 bis 4 der CPGTIO oder von 4 bis 7, je nachdem, ob vom leeren Bildschirm eine Trans-Id eingegeben wurde oder aus einem vorformatierten Feld.

CPGTIM Uhrzeit numerisch.

Dieses Feld enthält die Uhrzeit in numerischer (gepackter) Form in einem 6-stelligen Feld (HHMMSS) in Stunden, Minuten und Sekunden. Dieses Feld kann zum Ausrechnen von Zeitintervallen in den Rechenbestimmungen benutzt werden.Das Feld wird beim Programmaufruf und durch die Operation TIME aktualisiert. Da das Feld in der Methodenbank liegt, kann eine Aktualisierung auch von außen erfolgen. Falls diese nicht gewünscht wird, muss das Feld unmittelbar nach der Operation TIME mit Hilfe einer Z-ADD-Operation gerettet werden.

CPGTIS Terminal Id simuliert

CPG5-Anwendungen laufen nicht an einem Terminal ab. Zur eindeutigen Identifizierung der Browser-Session stellt CPG diese Id als 8-stelliges alphanumerisches Feld zur Verfügung.

CPGTSN Variabler Temporary Storage-Name

In diesem 8-stelligen Feld wird der Temporary-Storage-Name eingetragen, wenn in der Files Division variable Temporary-Storage-Verarbeitung vereinbart wurde.

CPGVRL Länge eines Satzes einer VSAM-Datei mit variabler Satzlänge

In diesem 5-stelligen numerischen Feld steht nach den Leseoperationen die Länge des eingelesenen Satzes. Für die Ausgabe kann das Feld genutzt werden, falls in der Satzbestimmung das Schlüsselwort VAR zusätzlich zu ALG und ADD angegeben wird. In diesem Fall wird die Länge des Ausgabesatzes dem Feld CPGVRL entnommen.

Vor der Verwendung von internen Feldern, die eine Verbindung zu bestimmten Bereichen des TP-Steuerprogramms sicherstellen, sollte der Programmierer prüfen, ob das von ihm verwendete TP-Steuerprogramm diese Bereiche zur Verfügung stellt.

Datum und Uhrzeit 2134

CPG stellt folgende interne Felder für Datum und Uhrzeit zur Verfügung:

CPGDAI enthält das Tagesdatum in der Form '0JJJJMMTTC'. (num)

Im Hinblick auf den Jahrtausendwechsel ist diese Datumsform sinnvoll, da sie durch die vollständige Angabe der Jahreszahl vergleichbar und durch den Aufbau sortierbar ist.

CPGDAT enthält das Tagesdatum in der Form '0TTMMJJJJC', also mit vierstelliger Jahreszahl. Als achtstelliges numerisches Feld kann CPGDAT mit Edit-Code 'Y' aufbereitet werden.

CPGTIM enthält die aktuelle Uhrzeit in der Form '0HHMMSSC'. Als sechsstelliges numerisches Feld kann CPGTIM mit dem Edit-Code Y aufbereitet werden. CPGTIM kann im Programm mit der Operation TIME aktualisiert werden.

UDATE enthält das Tagesdatum in der Form: 'TT.MM.JJ'. (alpha)

UDATEC enthält das Jahrhundert in der Form: 'JJ' (alpha)

UDATEI enthält das Tagesdatum in der Form: 'JJJJMMTT' (alpha)

UDAY enthält den Tag des Tagesdatums : '0TTC' (numerisch)

UMONTH enthält den aktuellen Monat: '0MMC' (numerisch)

UTIME enthält die Uhrzeit in der Form 'HH.MMUHR'. (alpha)

UYEAR enthält das aktuelle Jahr : '0JJC' (numerisch)

Speicherungsarten 2140

Alphanumerische Felder 2141

Alphanumerische Felder können alle druckbaren und nicht druckbaren Zeichen aufnehmen. Jedes Zeichen belegt ein Byte im Speicher. Alphanumerische Felder werden vor Beginn der Verarbeitung auf Blank initialisiert.

Numerische Felder 2142

Numerische Felder können nur Ziffern und gegebenenfalls ein Minuszeichen enthalten. Numerische Felder werden immer in gepackter Form gespeichert, das heisst in einem Byte werden jeweils 2 Ziffern untergebracht. Ein Halbbyte wird reserviert für das Vorzeichen. Numerische Felder werden vor Beginn der Verarbeitung auf Null initialisiert.

Datenfelder, mit denen Rechenoperationen durchgeführt werden sollen, müssen immer numerisch definiert sein.

Zur Vermeidung unnötiger Konvertierungen wird empfohlen, numerische Felder bei der Zwischenspeicherung auf Platteneinheiten ebenfalls in gepacktem Format auszugeben.

Natürlich ist auch die Ausgabe in ungepacktem Format möglich. In diesem Fall belegt jede Ziffer ein Byte. Das Vorzeichen wird zusammen mit der letzten Ziffer im rechtesten Byte des Feldes untergebracht. Dies kann dazu führen, dass diese Ziffer bei der Ausgabe in einen Buchstaben umgewandelt wird.

CPG gibt allen ungepackten numerischen Feldern, wenn sie positiv sind, automatisch ein Vorzeichen mit, das die letzte Ziffer wieder lesbar macht.

Achtung: Wenn bei der ungepackten Ausgabe numerische Felder teilweise durch andere Felder überlagert werden, muss diese Automatik durch Eintragung von 'SIGN' in der OPTIONS-Karte ausgeschaltet werden.

Binäre Felder 2143

Numerische Felder können binär ausgegeben und wieder eingelesen werden. Die binäre Ausgabe erfolgt durch Eintragung des Schlüsselworts 'BIN' in der Feldbestimmung der Output Division.

Beispiel : - NUMMER 15 BIN

Eingabefelder werden als binär interpretiert, wenn in der Feldbestimmung der Input Division vor der Position 'B' oder 'BIN' codiert wird.

Beispiel : - B 11 15 0 NUMMER

Für numerische Felder bis zu 4 Stellen werden bei der Ausgabe 2 Byte große Felder angelegt und für Felder bis zu 9 Stellen 4 Byte große Binärfelder.

Bei Bildschirm- und Drucker- -Ein- und -Ausgaben dürfen keine binären Felder benutzt werden.

Logisch gepackte Felder 2144

Neben der Möglichkeit, numerische Felder in gepacktem Format zu speichern, bietet CPG eine weitere Einsparung von Plattenspeicherkapazität durch die Verarbeitung logisch gepackter Felder an.

Ein logisch gepacktes Feld ist ein Feld, dessen numerischer Inhalt gepackt und ohne Vorzeichen gespeichert wird.

Beispiel:

Die Speicherung der Zahl 4711 erfordert:

Ungepackt	4 Bytes	F4 F7 F1 F1
Gepackt	3 Bytes	04 71 1C
Logisch gepackt	2 Bytes	47 11

Datenfelder, die logisch gepackt werden sollen, dürfen nur positive numerische Werte enthalten. Die logisch gepackte Form gilt nur für die Speicherung in externen Speichern. Zur Verarbeitung im Anwendungsprogramm werden die Daten wieder in das gepackte Format zurückübertragen.

Numerische Felder werden logisch gepackt ausgegeben,wenn bei der Feldausgabe das Schlüsselwort 'LOG' eingetragen wird.

```
-           NUMMER 15 LOG
```

Numerische Felder werden logisch gepackt eingelesen, wenn vor der ersten Eingabeposition ein 'L' eingetragen wird.

```
-           L 11 15 0 NUMMER
```

Ungepackte numerische Felder 2145

Numerische Felder werden immer im Hauptspeicher (TCA) gepackt. Durch das Einlesen vom Bildschirm erhält ein numerischer Wert deshalb intern im letzten Halbbyte den Buchstaben 'F' für die Zone.

Wird ein numerischer Wert aus dem Hauptspeicher auf Platte ungepackt ausgegeben, so wird die Zone grundsätzlich auf 'F' gesetzt. Der Wert 123 steht dann beispielsweise auf der Platte als F1 F2 F3.

Beim Einlesen von der Platte wird die Zone nicht geändert.

Diese Regeln für das Zonenhalbbyte des gepackten Feldes können durch den OPTIONS-Parameter SIGN beeinflusst werden.

Der Eintrag bewirkt, dass beim Einlesen sowohl vom Bildschirm als auch von der Platte grundsätzlich die Zone 'C' angenommen wird. Bei der Ausgabe auf Platte wird die Zone nicht verändert.

Bei ungepackter Ausgabe steht in diesem Fall die Zahl 123 auf Platte als F1 F2 C3, also 12C.

vorher	C	F	C	F
nach Bildschirm-Eingabe	F	F	C	C
nach Platten-Eingabe	C	F	C	C
nach Platten-Ausgabe	F	F	C	F
OPTIONS	ohne 'SIGn'		mit Parameter 'SIGn'	

Bei numerischen Operationen (Rechenoperationen) erhält das Ergebnisfeld unabhängig vom Options-Parameter immer die Zone C (bei positivem Ergebnis).

Bildschirm-Eingabefelder 2150

Im Gegensatz zu Eingabefeldern anderer Dateien wird ein Bildschirmeingabefeld mit einer MAP-Operation nur dann eingelesen, wenn das Feld vor der Operation am Bildschirm verändert wurde. Hardwareseitig wird dies dem Steuerprogramm durch ein 'Modified Data Tag' mitgeteilt. Das 'Modified Data Tag' kann jedoch auch vom Programmierer durch den Aufbereitungsschlüssel gesetzt werden. In diesen Fällen wird das Eingabefeld auch gelesen, wenn es vorher nicht verändert wurde.

- Ein einzulesendes Feld wird identifiziert durch die mit der Eingabenachricht übertragene Adresse des 1.Bytes dieses Feldes, d.h. es können nur solche Felder eingelesen werden, die vorher bereits einmal auf dem Bildschirm ausgegeben (vorformatisiert) wurden.

Bei numerischen Feldern wird die Dezimalstellenanpassung vom CPG wie folgt durchgeführt: Dezimalstellen werden ab dem ersten eingegebenen Komma oder nach der letzten gültigen Ziffer dem definierten Eingabefeld angepaßt.

Werden keine Dezimalstellen eingegeben,so werden nötigenfalls fehlende Dezimalstellen als Nullwerte eingefügt. (Siehe Beispiel 3 und 5). Zuviel eingegebene Dezimalstellen werden ignoriert, wie im Beispiel 2 zu sehen ist. Werden mehr Ziffern eingegeben als ein numerisches Feld aufnehmen kann, wird ab der angenommenen Kommastelle das Feld angepasst. Siehe hierzu auch die Beispiele 3 und 6 (nächste Seite).

Beispiel: FELDA wird numerisch definiert mit 4 Stellen 2 dezim.
FELDB wird numerisch definiert mit 5 Stellen 2 dezim.

```
-      110 113 2 FELDA
-      209 213 2 FELDB
```

Die Felder 'FELDA, und 'FELDB' sollen vom Bildschirm gelesen werden; dazu muss der Bildschirm vorher 'initialisiert', d.h. zur Aufnahme der Daten vorformatisiert werden. Dies erfolgt durch die vorherige Ausgabe eines entsprechenden Feldes auf den Bildschirm.

Dieses Feld muss in jedem Fall einen Aufbereitungsschlüssel mit der Feldeigenschaft 'ungeschützt' erhalten.

Beispiel	Feld		Pos.	Maske	Eingabe	Feldinhalt	Nummer
1		A	115	' ' '	'12,34 '	12,34	1
2					'1,2345'	1,23	2
3					'123 '	23,00	3
4		A	215	' ' '	'0,1234 '	0,12	4
5					'123 '	123,00	5
6					'1234567'	567,00	6
7	FELDA	A	116	' , '	' 4,25'	4,25	7
8	FELDA	A	118	' 0 , CR'	' 12,34CR'	12,34-	8
9	FELDB	A	215	' , '	' 4,258'	4,25	9
10	FELDB	A	217	' 0 , CR'	' 12,34CR'	12,34-	10

Zwischenspeicherung von Daten 2160

Online-Anwendungen erfordern mitunter eine stufenweise Verarbeitung der Daten durch verschiedene Programme, die nacheinander ablaufen. Da beim Aufruf eines neuen Programms die von diesem Programm benutzten Datenfelder in der Regel neu formatisiert, d.h. gelöscht werden, erfordert eine solche 'Verkettung' von Programmen eine Zwischenspeicherung der Daten in einem Bereich, der auch nach Beendigung eines Programms erhalten bleibt.

Die Zwischenspeicherung kann auf verschiedene Arten erfolgen:

1. Zwischenspeichern von Daten in der Transaction Work Area

Die Transaction Work Area (TWA) wird bei jedem Programmaufruf neu formatisiert. Jedoch kann der Programmierer in der Options Division hinter dem Schlüsselwort 'TWA' die Anzahl Bytes der TWA beginnend vom Anfang eintragen, die nicht neu formatisiert werden soll.

Bei dieser Form der Zwischenspeicherung sind die Regeln für Programmverbindungen zu beachten.

2. Zwischenspeichern von Daten in der Terminal-Tabelle.

Mit Hilfe der EDIT-Operation kann der Programmierer Daten in der TCT des Steuerprogramms abstellen, die zu beliebigen Zeitpunkten mit beliebigen Programmen, aber nur vom selben Terminal wieder abgerufen werden können. Der Abruf erfolgt mit Hilfe einer SELCT-Operation. Das Ergebnisfeld heisst in beiden Fällen 'CPGTCT' und braucht im Programm nicht definiert zu werden.

Beispiel:

```
Daten speichern:     - -C. EDIT  CPGTCT

                     - -O. FIELD CPGTCT.   WERT   20    PAC
```

Das numerische Feld 'WERT' wird in gepacktem Format in der TCT gespeichert. Die letzte Stelle des Feldes steht auf Position 20 des Feldes 'CPGTCT'.

```
Daten abrufen:       - -I. FIELD CPGTCT.  PAC  16 20 2  WERT

                     - -C. SELCT CPGTCT
```

Das in den Positionen 16-20 abgestellte numerische Feld wird in das Feld 'WERT' übertragen. Das Feld war in der TCT gepackt gespeichert. Das Feld 'WERT' wird mit 2 Dezimalstellen definiert.

Die Größe des Feldes 'CPGTCT' wird bei der Generierung der Terminal-Tabelle (TCT) festgelegt, und zwar muss der Parameter, der die User-Area Length beschreibt, um 10 Bytes größer sein als die höchste in einem Programm verwendete Ausgabe-Position. Im obigen Beispiel also Position 20 + 10 Bytes ergibt eine TCT User Area Length von mindestens

30 Bytes.

3. Zwischenspeichern von Daten in der CSA.

Mit der oben für die TCT beschriebenen Methode können auch Daten in der Common System Area (CSA) zwischengespeichert werden. Diese Daten können dann zu beliebigen Zeitpunkten mit beliebigen Programmen von beliebigen Terminals wieder abgerufen werden. Hierzu gilt das oben für die TCT beschriebene Beispiel, wenn anstelle von 'CPGTCT' der Feldname 'CPGCSA' eingesetzt wird.

Die Größe des Feldes 'CPGCSA' wird bei der Generierung der System-Initialisierungs-Tabelle des jeweiligen TP-Monitors festgelegt, und und zwar muss der CSA-Size-Parameter um mindestens 16 Bytes größer sein als die höchste in einem Programm verwendete Ausgabeposition, da CPG intern die ersten 16 Stellen benutzt.

4. Zwischenspeichern von Daten auf dem Bildschirm.

Der Bildschirm kann ebenfalls als Zwischenspeicher benutzt werden, wenn die Daten am Ende eines Programms dorthin ausgegeben werden, jedoch kann das Folgeprogramm diese Daten nur dann lesen, wenn bei der Ausgabe ein Aufbereitungsschlüssel mit der Feldeigenschaft 'Modified-Data-Tag-gesetzt' spezifiziert wurde und das Programm nach 'CPGEND' verzweigt.

5. Temporary Storage und Temporary Storage Queuing siehe Kapitel 2190.

6. Zwischenspeichern von Daten auf Transient-Data.

Sequentiell ausgegebene Daten können auf Transient Data zwischengespeichert werden. CPG verwendet Transient Data in jedem Fall zur Zwischenspeicherung von Drucker-Ausgaben. Der Programmierer kann diesen Service auch für eigene Anwendungen benutzen. Er sollte dabei jedoch die Auswirkungen auf die Performance übersehen können. Die Programmierung ist nicht anders als für Platten-Dateien. Die entsprechende Eintragung in der Dateizuordnung ist 'TRANSDT'.

7. Zwischenspeichern von Daten auf Platten-Dateien.

Bei zu kleinem Hauptspeicher (hohe Paging-Rate) sollte in Zweifelsfällen auch die Zwischenspeicherung von Daten auf Platten-Dateien erwogen werden, um gegebenenfalls den Hauptspeicher zu entlasten.

8. Zwischenspeichern in der Common Area (bei Command Level Programmen)

Nur im Command Level steht die Common Area für das Zwischenspeichern von Daten zur Verfügung. Die Common Area ist 4080 Bytes groß. CPG kann über das Feld CPGCOM und die Operationen EDIT und SELCT mit

der Common Area kommunizieren.

Die Länge der übergebenen Common Area kann durch SELECT CPGPIW ermittelt werden. Sie steht dort binär gepackt in den Stellen 239-240.

Zwischenspeichern von Daten auf Temporary Storage. 2190

Der Programmierer kann einen Bereich im Hauptspeicher oder in einem angeschlossenen Hilfsspeicher für die Zwischenspeicherung von Daten verwenden, soweit der verwendete TP-Monitor dies zulässt.

CPG2 ordnet ihnen einen symbolischen Namen (bis zu acht Zeichen) zu. Unter diesem Namen werden sie zwischengespeichert,aufgesucht und freigegeben. Der Name setzt sich wie folgt zusammen:

XXXXYYYY

X = Terminalkennzeichen; beim Zusatz 'I' zur FILE-Bestimmung in der OUTPUT-Division ist dieser Teil gleich '****'.
Y = Storagename aus der FILES-Division

Dazu muss dieser Bereich in einer Dateizuordnungskarte mit einem beliebigen 4-stelligen Namen definiert werden. Die Einheit (letzter Eintrag) lautet dabei 'STORAGE'. Die Länge des reservierten Speichers ergibt sich aus der Eintragung für die Satzlänge.

Außerdem bietet CPG2 die Möglichkeit, den Dateinamen variabel zu halten (Kapitel 3400). Bei der Programmausführung muss im Feld CPGTSN der Storage-Name (max. 8 Stellen) übertragen werden, der verarbeitet werden soll. Dies sollte nach Möglichkeit bei einer Non-Terminal-Task eingesetzt werden.

Um Schwierigkeiten zu vermeiden,die durch doppelt vergebene Namen entstehen können, sollten Benennungskonventionen eingeführt werden, an die sich die Programmierer halten müssen.

Ein temporärer Nachrichtenbestand sollte zum frühest möglichen Zeitpunkt freigegeben werden, damit für diesen Zweck nicht zu viel Speicher belegt wird. Für die Ein- und Ausgabebestimmungen gelten dabei die gleichen Regeln wie für Platten-Dateien.

Die Daten können sowohl als Informationseinheit als auch als untergliederter Nachrichtenbestand (Queue, siehe 2195) angelegt werden.

In der Satzbestimmung der Output-Division oder bereits in der Dateibestimmung der Files-Division kann der Programmierer durch Einträge bestimmen, ob der Satz im Hauptspeicher bleiben oder auf Platte ausgelagert werden soll und ob die Daten nur vom eigenen Terminal oder von allen Terminals gelesen werden können.

In der Output Division sind folgende Einträge hinter dem Storagenamen unterstützt:

Blank Die Daten können nur vom eigenen Terminal gelesen werden und werden im Hauptspeicher gespeichert.

I Die Daten können von allen Terminals gelesen werden und werden im Hauptspeicher gespeichert (Independent Storage)

A Die Daten können nur vom eigenen Terminal gelesen werden und werden im Hilfsspeicher gespeichert. (Auxiliary Storage)

In der Files Division kann vor der Einheit STORAGE eines der Schlüsselwörter AUX für Auxiliary Storage und IND für Independent

Storage angegeben werden. Diese Schlüsselwörter haben die gleiche Wirkung wie A und I in der Output Division.

Einträge in der Output Division haben Vorrang vor den entsprechenden Einträgen in der Files Division.

In den Rechenbestimmungen können temporär gespeicherte Daten mit einer READ-Operation gelesen werden. Dabei kann der Programmierer mit der Servicefunktion der READ-Operation bestimmen, ob der Bereich freigegeben oder für spätere READ-Operationen wieder verwendet werden soll. Eintragung der Service-Funktion:

Blank Der Bereich wird nach dem Lesen freigegeben.
'SAVe' Der Bereich wird nach dem Lesen nicht freigegeben.

Der Bereich wird jedoch in jedem Fall wieder freigegeben, wenn eine erneute Ausgabe unter demselben Namen erfolgt.

Bei Temporary Storage Queuing muss das Löschen mit dem expliziten Befehl PURGE durchgeführt werden.

Temporary Storage Queuing 2195

Es gibt fünf Verarbeitungsmöglichkeiten:

1. Eine Queue füllen

2. Einzelne Elemente verändern

3. Sequentiell lesen

4. Direkt lesen

5. Ganze Queue löschen

Bei Temporary Storage Queuing muss in der Dateibeschreibung der Files-Division ein 'Q' eingetragen werden (vgl. Kapitel 3400).

Für jede Queue wird intern ein CPGQxx-Feld 5-stellig numerisch angelegt. xx ist die Nummer der Dateizuordnung.

Als Satzformat muss immer FIX eingetragen werden, variable Satzlängen sind nicht unterstützt und werden wie feste Satzlängen behandelt.

Funktion 1 - Queue füllen:

- -I. FILE STOR DD

- -C. WRITE STOR

Dieses Beispiel zeigt, wie eine Queue gefüllt wird.

Es können maximal 32.000 Sätze in einer Queue gespeichert werden,

vorausgesetzt der TP-Monitor oder die Größe der Batch Partition lassen dies zu. Der Benutzer ist bei einer Überschreitung verantwortlich, es erfolgt keine Fehlermeldung.

Nach einer Ausgabe ist im Feld CPGQxx die relative Positionsnummer verfügbar.

Funktion 2 - einzelne Elemente verändern:

```
- -D.
-      LFNR  5 0
- -I.
-      FILE STOR
-         1 100  SATZ
- -C.
-      LFNR =  4
-      LFNR READ STOR
-      IF CPGFRC = '  '
-      LFNR UPDATE STOR
-      ENDIF
```

Dieses Beispielprogramm zeigt, wie einzelne Elemente einer Queue verändert werden. Wird ein ausgewähltes Element nicht aufgefunden, so wird die Bedingung EF (End of file) gesetzt.

In diesem Beispiel wird das vierte Element verändert, bei 'Not found' wird die Ausgabe nicht durchgeführt.

Hinweis: Im Gegensatz zu der UPDATE-Funktion bei Dateien ist bei der Ausgabe auf Temporary Storage der Ausgabebereich generell auf Blank gelöscht. Soll der ursprüngliche Satz bei der Ausgabe erhalten bleiben, so ist er aus der Eingabe in die Ausgabe zu übertragen. In dem Beispiel werden dann die Stellen 1 - 6 des Satzes durch die Konstante 'UPDATE' und die Stellen 98-100 durch die gepackte LFNR überschrieben.

Funktion 3 - sequentiell lesen (numerische Konstante möglich):

```
         1 READ STOR.                                *   1. Satz
         DO UNTIL CPGFRC = 'EF'                      *   Folgesätze
            READ STOR
         ENDDO
```

Dieses Beispielprogramm zeigt, wie eine Queue sequentiell gelesen wird.

Mit einer READ-Operation ohne Faktor 1 werden die Elemente der Reihe nach sequentiell gelesen. Um jedoch bei einem bestimmten Satz (z.B. Satz 1) mit dem Lesen zu beginnen, ist dieser Satz gezielt mit einem definierten Schlüssel zu lesen.

Funktion 4 - direkt lesen:

```
    -D.
        KEY 5 0
    -C.
        KEY = 5
        KEY READ STOR
```

Dieses Beispiel zeigt, wie aus einer Queue direkt gelesen wird. Es wird mit dem in Faktor 1 angegebenen Schlüssel zugegriffen, um den entsprechenden Satz zu lesen. Anders als bei Dateioperationen muss der Programmierer bei den folgenden READs dafür sorgen, dass im Key jeweils die gewünschte Satznummer enthalten ist, da mit 'KEY READ STOR' immer eine direkte Verarbeitung beschrieben wird. Daher können die Operationen CHAIN, RNDOM, SETLL usw. hier nicht verwendet werden.

```
- -C.
-      KEY  = 0
-      DO UNTIL CPGFRC = 'EF'
-         KEY = KEY + 1
-         KEY READ STOR
-      END
```

Dieses Beispiel zeigt, wie eine gesamte Queue direkt gelesen werden kann.

Funktion 5 - Queue löschen:

```
-      PURGE  STOR
```

Mit der Operation PURGE wird die in Faktor 2 definierte Queue gelöscht.

Beim Löschen einer Storage Queue wird der belegte Bereich freigegeben und dem verfügbaren dynamischen Speicherbereich zugeschlagen.

Es gibt keine Möglichkeit, nur ausgewählte Sätze aus einem Nachrichtenbestand freizugeben; daher ist beim READ eine Angabe für eine Service-Funktion nicht sinnvoll. Alle Sätze bleiben solange erhalten, bis der Bereich mit PURGE gelöscht wird.

Simulation des Queuing 2198

Mit der Eintragung S in der Files Division bzw. im Data Dictionary erreicht man, dass ein TS-Bereich vom Programmierer nach der Einzelsatzlogik verarbeitet werden kann, dass vom CPG aber eine Queue generiert wird, die aus einem Satz besteht.

Beispiel:

```
    - FILE STOR UPD  S  FIX  256  STORAGE.
```

Blockdiagramm 2240

Standardmäßig wird rechts neben der Umwandlungsliste ein Blockdiagramm des CPG2-Programms angelistet, das das Lesen des Programms auch für den Außenstehenden erleichtern soll.

Das Blockdiagramm hat folgenden Aufbau:

```
Blockdiagramm       PhaseX      Überschrift gemäß OPTIONS

Bildschirm                      Dateizuordnung L3270
Platte                          Dateizuordnung DISK

DATUM  ---------   11   16      Data Division, TWA-Überlagerung
       TAG         11   12
       MONAT       13   14
       JAHR        15   16

INPUT  -------------PLATTE      Eingabedatei INPUT (Plattendatei)
       X              75        Feld X ist 75 Stellen lang (ALPHA)
       Y               7,2      Feld Y ist 7 Stellen,davon 2 Dez.
       FG      10 *    5        Feldgruppe 10 * 5 Stellen lang

--------------------------      Beginn der Rechenbestimmungen

ANFANG-------I                  ANFANG TAG
            I                   Rechenbestimmung ohne Verzweigung
           IFI                  EDIT- oder SELCT-Operation
01 ---      I                   Schachtelungstiefe für DO- und IF-
            I                   Gruppen ( bei DO, IF und END )
02 ---      I
 02 --      I                   eingerückt bei ELSE, BREAK, CONTINUE
02 ---      I
01 ---      I
           /// ALL              EXCPT
            I======UPRO         EXSR Subroutine  oder EXPR Unterprogr.
            I
            I------OUT          GOTO OUT oder EXITP OUT
   INPUT   ///                  Lesen Datei INPUT
            I
           < >-----ANFANG       GOTO ANFANG wenn... (z.B. 15 ein)
            I
           /// OUTPUT           UPDATE, ADD ( WRITE ), OUTPUT
            I
            -------ANFANG       GOTO Anfang ohne Bedingung

--------------------------      Ende der Rechenbestimmungen

OUTPUT  ------------------      Ausgabedatei OUTPUT
   X        1        75         Feld X beginnt auf St. 1, ist 75 lang
   Y       76         7,2       Feld Y beginnt auf 76 ist num 7St/2Dez
   FG       1 10 *    5         Feldgruppe beginnt auf 1 ist 10*5 lang
```

Zu beachten ist, dass nur jeweils das erste Statement einer Zeile beim Blockschaltbild berücksichtigt wird. Soll also das Blockschaltbild genutzt werden, so ist dies bei der Codierung zu berücksichtigen.

Blockdiagramm bei Überlagerungen mit ORG:

Die flexibelsten Möglichkeiten zur Redefinition und Überlagerung in der Data Division bietet der Befehl ORG. In der Data Division gibt das Blockdiagramm pro Feld die Von- und die Bis-Position an. Beim ORG-Befehl wird stattdessen angezeigt: 1. die nächste Von-Position, wenn kein ORG codiert worden wäre und 2. die neue Von-Position als Ergebnis des ORG-Befehls.

Beispiel: Source-Code				BLOCKDIAGRAMM		PHASE	
D	F1		10		F1	1	10
D	F2	5	10		F2	11	60
D	F3		5		F3	61	65
D	F4		5		F4	66	70
D	F3	ORG		ORG	F3	71	61
D	F5		5		F5	61	65
D	F5	ORG		ORG	F5	66	61
D	F6		5		F6	61	65
D		ORG		ORG		66	71
D	F7		10		F7	71	80

Ausnahmen von den allgemeinen Syntaxregeln gelten für die Eintragungen zur Anpassung der Umwandlungsliste:

/EJECT für den Seitenvorschub an beliebiger Stelle
/NOLIST zum Unterdrücken des Listing
/LIST zum Aufheben von /NOLIST

Diese Befehle erfordern keinen Eintrag in Spalte 6 und beginnen immer auf Spalte 7.

Programmierer-Checkliste 2246

Im Anschluss an die Umwandlung wird eine Programmierer-Checkliste gedruckt, die den Programmierer auf bestimmte Voraussetzungen des TP-Steuer-Programms hinweisen und Auskunft über die wichtigsten Programmdaten geben soll.

Die Checkliste hat folgenden Aufbau:

```
TITEL                                   KD-NR KURZNAME       26.02.93
_____________________________________________________________________

        PROGRAMMIERER CHECKLISTE

              ENTHÄLT DIE CICS PCT EINE EINTRAGUNG MIT DEM PARAMETER
                                     PROGRAM=CPGTST UND
                                     TWASIZE=0166 ODER GRÖßER

              ENTHÄLT DIE CICS PPT EINE EINTRAGUNG MIT DEM PARAMETER
                                     PROGRAM=CPGTST

              ENTHÄLT DIE CICS FCT EINE EINTRAGUNG MIT DEM
                                     DATEINAMEN PLATTE


       PROGRAMMDATEN

              PROGRAMMGRÖSSE = 3.500 BYTES (CIRCA)

              TWA GRÖSSE     =   166 BYTES

              TIOA GRÖSSE    =     4 BYTES

              DEFINIERTE SCHALTER

              01 17 25 C1 T3 P2 DE


       EXTERNE PROGRAMMVERBINDUNGEN

              EXIT:  'TRID'
              EXIT:  PHASEX
              EXPR:  PHASEY
              PROG:  QPGMODUL                                          *
              EXHM:  MODUL
```

Die Eintragungen in den Programm- und Datei-Steuertabellen gelten für andere TP-Steuerprogramme sinngemäß.

Die Angabe der Programmgröße ist eine Circa-Angabe. Die exakte Programmgröße kann durch Umrechnung der bei 'CPGPND' angegebenen Adresse ermittelt werden.

Die TWA-Größe ist die Größe des erforderlichen Arbeitsspeichers. Sie wird exakt gerechnet. In der TWA-Größe sind jedoch vom Benutzer eingefügte Copy-Books nicht enthalten. Eine Kontrolle ist auch immer dann erforderlich, wenn die Daten-Ein- und -Ausgabe über Datasets erfolgt, da unterschiedliche Datenbank-Systeme unterschiedliche Bereiche in der TWA belegen. Die TWA-Größe errechnet sich aus der Adresse bei CPGTND minus der Länge der TCA, die normalerweise 256 Bytes groß ist.

Die Terminal-IO-Area (TIOA) darf maximal 4080 Bytes groß sein. Eine größere TIOA führt zu Programmfehlern. Wenn der Wert das Maximum übersteigt - dies kann vorkommen, wenn viele kleine Felder auf den Bildschirm ausgegeben werden - ist die Terminal-Ausgabe zu teilen.

Alle im Programm verwendeten Schalter werden in sortierter Reihenfolge in der Checkliste angelistet.

Alle externen Programmverbindungen werden am Ende der Checkliste in der beschriebenen Form angelistet. Der Programmierer muss dabei sicherstellen, dass bei Verwendung der EXITP oder EXPR Operation mit dem Phasennamen in Faktor 2 keine Schleifen entstehen, da diese zu erheblichen Performance-Verlusten führen können.

Cross Reference 2247

Im Anschluss an die Umwandlung kann auf Wunsch eine Cross-Reference-Liste ausgedruckt werden. Hierzu ist der OPTIONS-Parameter XREF einzutragen.

Die Cross-Reference-Liste gibt Auskunft darüber, bei welchen CPG-Statement-Nummern die folgenden Programm-Elemente verwendet wurden:

- Dateinamen
- Bezugszahlen und Anzeiger
- Alphanumerische Konstanten
- Datenfelder, Feldgruppen und Tabellen
- Numerische Konstanten
- Sprungmarken, interne und externe Unterprogramme
- Operationen

Schalter 2260

Der Ablauf eines CPG-Programms kann mit Hilfe von Bezugszahlen gesteuert werden. Die Bezugszahlen 01 bis 99 können vom Programmierer gesetzt oder gelöscht werden. Diese Programmierweise kann als veraltet angesehen werden und wird hier nicht weiter beschrieben.

Schalter (wie die unten beschriebenen) sollten mit der Operation IF CONDITION abgefragt werden.

Folgende Schalter werden vom Terminal-Benutzer zur Ausführungszeit des Programms durch Tastendruck gesetzt und können im Programm abgefragt werden:

P1, F1 oder PF1	Programmfunktionstaste 1
P2, F2 oder PF2	Programmfunktionstaste 2
P3, F3 oder PF3	Programmfunktionstaste 3
P4, F4 oder PF4	Programmfunktionstaste 4
P5, F5 oder PF5	Programmfunktionstaste 5
P6, F6 oder PF6	Programmfunktionstaste 6
P7, F7 oder PF7	Programmfunktionstaste 7
P8, F8 oder PF8	Programmfunktionstaste 8
P9, F9 oder PF9	Programmfunktionstaste 9
PA, F10 oder PF10	Programmfunktionstaste 10
PB, F11 oder PF11	Programmfunktionstaste 11
PC, F12 oder PF12	Programmfunktionstaste 12
Q1, F13 oder PF13	Programmfunktionstaste 13
Q2, F14 oder PF14	Programmfunktionstaste 14
Q3, F15 oder PF15	Programmfunktionstaste 15
Q4, F16 oder PF16	Programmfunktionstaste 16
Q5, F17 oder PF17	Programmfunktionstaste 17
Q6, F18 oder PF18	Programmfunktionstaste 18
Q7, F19 oder PF19	Programmfunktionstaste 19
Q8, F20 oder PF20	Programmfunktionstaste 20
Q9, F21 oder PF21	Programmfunktionstaste 21
QA, F22 oder PF22	Programmfunktionstaste 22
QB, F23 oder PF23	Programmfunktionstaste 23
QC, F24 oder PF24	Programmfunktionstaste 24

A1 oder PA1	Programmabruftaste	PA1	Bei der Verwendung von PA-
A2 oder PA2	Programmabruftaste	PA2	Tasten werden keine Daten
A3 oder PA3	Programmabruftaste	PA3	vom Bildschirm eingelesen.

DE	Daten-Freigabe
SP	Lichtstift-Abfrage / Cursor-Select-Taste (Pos. Ausw.)
CL o. CLEAR	Löschtaste

Beachte: Diese Tasten können im Programm auch in dem internen Feld CPGMPF abgefragt werden (zweistellig, also P1 bis QC etc.).

Diese Schalter sowie 'NI' und 'IC' (s.u.) bleiben auch bei Programmverbindungen bis zur nächsten Bilschirmeingabe erhalten.

Bei der Verarbeitung von Plattendateien werden die folgenden Schalter gesetzt, wenn die entsprechende Bedingung auftritt. Die Schalter können vom Programmierer unmittelbar nach der Ein-/ Ausgabe-Operation abgefragt werden.

EF oder EOF	Dateiende bei Transient Data (Queue Zero)
EF oder EOF	Dateiende bei sequentieller Plattenverarbeitung
EF oder EOF	Dateiende oder Fehler bei Temporary Storage Queues
EF oder EOF	Wenn zum Hinzufügen bei Temporary Storage Queues kein Platz mehr vorhanden ist.
EF oder EOF	Datei-Anfang bei READ-BACK (nur VSAM)
EF oder EOF	Wenn bei Programmaufrufen mit EXITP Programmname oder EXPR das angegebene Programm nicht in der PPT liegt.
EF oder EOF	Wenn bei Programmaufrufen mit EXITI entweder die aufgerufene Task oder der angesprochene Bildschirm nicht in der entsprechenden CICS-Tabelle liegt.
DR	Doppelter Satz beim Hinzufügen.
DR	Doppelter Schlüssel bei READ Alternativindex.

Bei sequentieller Plattenverarbeitung wird vom CPG der Schalter 'EF' (End of File) gesetzt, wenn der letzte Satz der Datei gelesen wurde. Dieser Schalter kann vom Programmierer ebenfalls nur abgefragt werden.

Beim Hinzufügen von Sätzen zu einer ISAM- oder VSAM-Datei wird der Schalter 'DR' gesetzt, wenn ein Satz mit gleichem Schlüssel bereits vorhanden ist. Der Schalter wird bei der nächsten 'ADD'-Ausgabe wieder gelöscht.

Tritt der Fehler 'DR' beim sequentiellen Lesen eines Pfades einer VSAM Datei auf, so bedeutet dies, dass mehrere Sätze mit gleichem Key vorhanden sind. Wird der letzte Satz einer Gruppe gelesen, so wird der DR Schalter gelöscht.

Beachte: Diese File Return-Codes werden dem Programm auch im internen Feld CPGFRC zur Verfügung gestellt.

Nach einer Bildschirmeingabe kann der Schalter 'IC'(Invalid Character) abgefragt werden, der anzeigt, ob bei der Eingabe in eines der eingelesenen numerischen Felder ein nicht numerisches Zeichen gefunden wurde (z.B. Buchstabe o für Null). Der Schalter muss jedoch unmittelbar nach der READ- oder EREAD-Operation abgefragt werden.

Nach einer Bildschirmeingabe kann der Schalter 'NI' (No Input) abgefragt werden. Werden vom Bildschirm keine Daten eingelesen, so ist der Schalter NI gesetzt. Der Schalter muss jedoch unmittelbar nach der Lese-Operation abgefragt werden.

Nach einer Ausgabe mit EXCPT kann der Schalter 'NO' (No Output) abgefragt werden. Der Schalter wird gesetzt, wenn der EXCPT-Befehl nicht zu einer Ausgabe führt.

Ist Kanal 12 bei der List-Ausgabe erreicht, wird der Schalter 'OF' für Overflow gesetzt.

Im Batch stehen zusätzlich die UPSI-Schalter U1 bis U8 zur Verfügung.

Der Inhalt von Bezugszahl '00' hat folgende Bedeutung:

X'F0' Sequentielles Lesen
X'08' EF Bedingung
X'04' DR Bedingung
X'02' CHAIN U
X'01' ETC, Shadow File busy

Diese Werte können zwar im Programm nicht angesprochen werden, sind aber im Dump zu sehen.

Gruppenstufen 2270

Gruppenstufen (L0 bis L9) werden weitgehend wie beim RPG unterstützt, jedoch kennt CPG keine Detail- oder Totalausgaben. Ausgaben zur Gruppenzeit werden durch eine EXCPT-Anweisung aus den Total-Rechenbestimmungen verursacht.

Inkompatibilitäten zu RPG sind auf der Folgeseite aufgeführt.

Anders als beim RPG erfolgt der Gruppenwechsel nicht vor Beginn der Rechenbestimmungen, sondern unmittelbar bei der READ- oder CHAIN-Operation für eine Datei, d. h. die Rechenbestimmungen werden an dieser Stelle bei einem Gruppenwechsel zur Durchführung der Gruppenberechnungen unterbrochen. Die READ- oder CHAIN-Operation wirkt dabei wie die Operation EXSR. In den Total-Rechenbestimmungen darf kein Dateizugriff mit Gruppenstufenschalter durchgeführt werden.

Gruppenstufen können bei Platten-Dateien und bei Datasets verwendet werden. Ebenso können Gruppenstufen bei der Batch-Ausführung auch bei READER, TAPE und bei sequentiellen Plattendateien verarbeitet werden. Gruppenstufenschalter sind bei HL1-Dataset-Verarbeitung nicht unterstützt.

Der Gruppenwechsel ist kompatibel zur RPG Logik. Bei End-Of-File werden alle Gruppenschalter L1-L9 angesetzt und die Total Rechenbestimmungen durchlaufen. Vor der Verarbeitung des ersten Satzes erfolgt keine Total-Verarbeitung.

Beispiel für Gruppenstufen:

```
 1 - -I. FILE PLATTE.
 2 -        1  7 KDNR   L2
 3 -        8 15 AUTRAG L1
 4 -       16 80 SATZ

 5 - -C. LESEN
 6 -        DEBUG
 7 -        KDNR  READ  PLATTE
 8 -        ON EF  GOTO ENDE
 9 -        ON L1  ANZPOS = 0
10 -        ON L1  ANZAUF = ANZAUF + 1
11 -        ANZPOS = ANZPOS + 1
12 -        DEBUG
13 -        MAPD BILD1.              * SATZ anzeigen
14 -        GOTO LESEN
15 -     ENDE.
16 - -L1. DEBUG
17 -      MAPD BILD2.                * Anzahl Positionen anzeigen
18 - -L2. MAPD BILD3.                * Anzahl Aufträge anzeigen
```

Verfolgen wir den Ablauf des Programms mit Hilfe der programmierten DEBUG Operationen, so erhalten wir folgendes Ergebnis:

STMT	L1	L2	01	02	03	KDNR	AUFTRAG	ANZPOS	ANZAUFT	SATZ
6								0	0	
12	L1	L2				004711	08150001	1	1	1
6	L1	L2				004711	08150001	1	1	1
12						004711	08150001	2	1	2
6						004711	08150001	2	1	2
16	L1					004711	08150001	2	1	2
12	L1					004711	08150002	1	2	3
6	L1					004711	08150002	1	2	3
12						004711	08150002	2	2	4
6						004711	08150002	2	2	4
16	L1	L2				004712	08150002	2	2	4
12	L1	L2				004712	08150003	1	1	5
6	L1	L2				004712	08150003	1	1	5
12						004712	08150003	2	1	6

Unterschiede zum RPG:

1. In einem Verarbeitungsschritt kann ein Gruppenstufenschalter nur einem Feld zugeordnet werden, d.h.

   n i c h t möglich ist (wegen L2)

```
 - FILE DATEI
 -                                          3   8 FELD1 L2
 -                                         12  14 FELD2 L2
 -                                         15  17 FELD3 L1
```

   möglich ist natürlich

```
 - FILE DATEI
 -                                          3   8 FELD1 L2
 -                                         15  17 FELD3 L1
 - FILE DATEI2
 -                                          9  14 FELD2 L2
 -                                         15  17 FELD3 L1
```

2. Werden Gruppenstufenschalter bei mehreren verschiedenen Feldern angewandt, so müssen diese Felder die gleichen Feldeigenschaften haben, also gleich lang sein und vom gleichen Feldtyp (alpha / numerisch) sein.

   N i c h t möglich ist (wegen unterschiedlicher Länge bei L2)

```
 - FILE DATEI
 -                                          3   8 FELD1 L2
 -                                         15  17 FELD3 L1
 - FILE DATEI2
 -                                         12  14 FELD2 L2
 -                                         15  17 FELD3 L1
```

3. Zur Kennzeichnung der Total-Rechenbestimmungen werden die Stufen L0 bis L9 wie Division Indicator eingesetzt (vgl. Zeile 16 und 18).

   'Total LR' kann aber dennoch genutzt werden, indem man stattdessen Schalter L9 abfragt (vorausgesetzt, dass L9 im Programm ohne sonstige Funktion ist).

4. N i c h t unterstützt ist der Einsatz von Gruppenstufenschaltern in Kombination mit der Prüfung von Zeichen im Datensatz.

   Die im folgenden beschriebene Verarbeitung führt deshalb zu nicht vorhersehbaren Ergebnissen für die Schalter L1 und L2, weil jeweils nur eines der beiden Felder eingelesen wird.

```
 - FILE DATEI  02    2 C2
 -                                    1   5 KEY   L2
 - FILE DATEI  01    2 C1
 -                                   21  25 KDNR  L1
```

Dateiverarbeitung 2300

Dateinamen 2301

Dateinamen sind maximal 8 Stellen lang. Das 1. Zeichen muss ein Buchstabe von A bis Z oder ein '$' (Dollar) sein. Die folgenden Stellen können Ziffern und Buchstaben enthalten.

Dateinamen dürfen nicht mit Feldnamen übereinstimmen. Die Prüfung für eine CPG-Fehlermeldung kann nur für die ersten sechs Stellen des Dateinamens durchgeführt werden.

In einem Programm dürfen maximal 100 Dateien definiert werden.

Schlüssel 2304

Bei den Operationen CHAIN, DELET, READ, READP, UPDAT, SETLL und WRITE muss in der Regel ein Schlüsselfeld oder Schlüsselwert im Faktor 1 der Procedure Division angegeben werden.

Ein alphanumerischer Schlüsselwert kann als in Hochkommata eingeschlossene Konstante spezifiziert werden. Eine solche Konstante kann maximal 8 Bytes lang sein. Ist eine angegebene Konstante kürzer als die für die Datei in den Dateizuordnungen definierte Schlüssellänge, so wird der Rest mit X'00' formatisiert.

Hat der Schlüssel numerisches (d.h. gepacktes) Format, so kann der Schlüsselwert nicht als dezimale Konstante angegeben werden.

Wenn der Schlüssel als in der TWA definiertes Feld angezogen wird, sollte die Länge des Feldes der für die Datei in den Dateizuordnungen definierten Länge entsprechen. Ist es kürzer als die Schlüssellänge, so werden die restlichen Bytes mit X'00' formatisiert. Ist es länger als die Schlüssellänge, so wird es abgeschnitten (d. h. rechtsbündige Bytes werden ignoriert).

Wenn Dateien mit Schlüsselfeldern im gepackten Dezimalformat verarbeitet werden, muss mit Vorsicht vorgegangen werden. Für die IBM-Hardware gelten die Hexadezimalwerte 'C' und 'F' bei der Verarbeitung numerischer Felder als positives Vorzeichen. Dies kann zu einem scheinbaren Nichtauffinden des Satzes führen, wenn im Dateischlüssel das Vorzeichen 'F' codiert ist und im Programm ein Feld mit dem Vorzeichen 'C' spezifiziert ist. Das Vorzeichen des Schlüsselfeldes kann mit der MLLZO-Operation korrigiert werden.

Ist das Vorzeichen des Satzschlüssels in der Datei ein 'C' und steht im Programmfeld ein 'F' (wenn es z.B. von einem Bildschirm in ein numerisches Feld eingelesen wurde),kann das Vorzeichen des Programmfeldes mit Hilfe einer Z-ADD-Operation in ein 'C' umgewandelt werden.

Zusammengesetzte Schlüsselfelder können mit Hilfe der EDIT-Funktion oder durch Definition von Overlay-Feldern oder durch Datenstrukturen aufgebaut werden.

Sequentieller oder wahlfreier Zugriff 2306

Eine in einem CPG-Programm definierte DISK-Datei kann entweder wahlfrei (d. h. für jeden gelesenen Satz wird ein eindeutiger Schlüssel angegeben) oder sequentiell verarbeitet werden. Die Wirkung der CPG2-Dateioperationen hängt bis zu einem gewissen Grade davon ab, wie eine Datei verarbeitet wird.

Zu Beginn einer Transaktion wird davon ausgegangen, dass alle für das Programm definierten DISK-Dateien wahlfrei (RANDOM) verarbeitet werden sollen. Die in diesem Modus unterstützten Dateioperationen sind CHAIN, WRITE, UPDAT, DELET, EXCPT und RNDOM.

Wenn eine READ-, READP- oder READB-Operation für eine DISK-Datei ausgeführt wird, wird automatisch für diese Datei sequentieller Verarbeitungsmodus gesetzt.

In diesem Modus werden die Dateioperationen READ, READP, READB, SETLL, und CHAIN unterstützt.

Sobald eine Datei auf den sequentiellen Modus umgestellt worden ist, bleibt sie darin, bis eine RNDOM-Operation für diese Datei ausgeführt wird. Diese bewirkt, dass die Datei wieder auf den wahlfreien Zugriff umgeschaltet wird.

Dateioperationen im sequentiellen Zugriff 2307

Die sequentielle Verarbeitung einer DISK-Datei wird durch Ausführung einer READ-, READB-, (nur VSAM) oder READP-Operation eingeleitet.

Die Datei verbleibt im sequentiellen Modus,bis die wahlfreie Verarbeitung für die Transaktion mit Hilfe einer RNDOM-Operation wieder hergestellt wird.

Diese Leseoperationen dienen zur sequentiellen Verarbeitung einer ganzen Datei oder bestimmter Teile einer Datei (sprungsequentielle, generische Verarbeitung oder Suchlauf).

Mit der READB-Operation wird eine VSAM-Datei in Schlüsselfolge rückwärts abgearbeitet (d.h. in umgekehrter Schlüsselfolge).

Der Schlüsselwert dient lediglich zur Positionierung der Datei beim Lesen des ersten Satzes, wenn die sequentielle Verarbeitung gestartet wird. Bei einer READ- oder READP-Operation ist dies der erste Satz in der Datei, dessen Schlüssel gleich oder größer als der angegebene Schlüssel ist. Bei einer READB-Operation ist dies der Satz, dessen Schlüssel gleich dem spezifizierten Schlüssel ist. Ist der Schlüsselwert größer als der des letzten Satzes in der Datei (READ, READP) oder kleiner als der des ersten Satzes in der Datei (READB), so wird der Dateiende-Anzeiger (EF) 'EIN' gesetzt.Dieser muss unmittelbar nach der Leseoperation abgefragt werden.

Sobald die sequentielle Verarbeitung eingeleitet ist, lesen alle nachfolgenden Leseoperationen den nächstfolgenden Satz in der Datei. Dies geschieht ohne Rücksicht auf den tatsächlich angegebenen Schlüsselwert. Die Eintragung für das Schlüsselfeld muss jedoch (außer bei Batch-Programmen) vorgenommen werden.

Wenn das Dateiende festgestellt wird, wird der EF-Anzeiger 'EIN' gesetzt, und die Eingabebestimmungen für die Datei werden nicht ausgeführt. Das Programm wird fortgesetzt mit der nächsten Anweisung in den Rechenbestimmungen hinter der Leseoperation. Werden für die Datei weitere sequentielle Leseoperationen ausgeführt, so werden keine Sätze gelesen und der EF-Anzeiger wieder 'EIN' gesetzt.

Bei VSAM muss die EF-Bedingung unbedingt abgefragt werden. Ein weiteres READ führt zu einer Systemfehlermeldung.

VSAM Dateien, die im CICS mit READ, READB oder CHAIN U/P gelesen werden, belegen einen VSAM String, solange die TASK besteht, oder bis der String mit RNDOM freigegeben wird.

VSAM - Alternativindizes 2315

Eine VSAM-Datei kann mit einem Alternativschlüsselfeld gelesen werden. Dieses Leseverfahren mit Alternativschlüsseln erfordert einen weiteren Eintrag in der Dateitabelle (FCT).

Wird ein Satz im Basiscluster hinzugefügt und existiert der Alternativschlüssel (unique) bereits, so wird der Schalter DR (Duplicate Record) angesetzt.

Der DR-Anzeiger wird 'EIN' gesetzt, wenn eine Datei mit nicht eindeutigem Schlüssel gelesen wird.

Bei einem nachfolgenden Lesen mit eindeutigem Schlüssel wird der DR-Anzeiger 'AUS' gesetzt.

VSAM - ESDS/RRDS 2316

ESDS- und RRDS-Dateien werden über die relative Byte-Adresse (RBA) bzw. die relative Satznummer angesprochen. Für den Zugriff muss die RBA als binäres numerisches Feld in der Länge 9 oder als vierstelliges Alphafeld zur Verfügung stehen.

Wird bei einem Dateizugriff ein numerisches Keyfeld eingetragen,so erfolgt die binäre Aufbereitung automatisch. Wird ein Alpha-Schlüssel eingetragen, so ist der Programmierer selbst für die binäre Aufbereitung verantwortlich (vgl. Beispiele 5 - 7, Kapitel 8000)

VSAM - Dateien in Zugangsfolge 2317

Dateien in Zugangsfolge (auch eingabesequentielle Dateien genannt; ESDS – Entry Sequenced Data Sets) werden unterstützt. Wenn einer Datei in Zugangsfolge Sätze hinzugefügt werden, werden diese an das Ende der Datei angehängt. Ein Schlüssel bei einer EXCPT- oder UPDAT Operation wird ignoriert.

Wenn ein Satz gelesen wird, muss die relative Byteadresse (RBA) des Satzes als Schlüssel angegeben werden. Die RBA wird als Summe der Bytes sämtlicher zuvor aus dieser Datei gelesenen Sätze berechnet. Dies ist nur bei Sätzen fester Länge ohne weiteres möglich.

In der Dateizuordnung muss ein 'R' oder 'RBA' codiert sein, um die relative Byteadressierung zu definieren.

Um sicherzustellen, dass die hinzugefügten Sätze auch tatsächlich auf Platte gespeichert werden, sollte nach einem ADD oder WRITE der aktuelle Satz mit einer CHAIN-Operation gelesen werden. (Bei der Operation CHAIN wird das VSAM CI zurückgeschrieben.) Als Schlüssel wird das Feld CPGKxx angegeben, wobei xx für die Position der Datei in der Files Division steht.

DL/I 2330

Das Datenbank-System DL/I wird über ein CPG-DL/I-Interface für Online-Verarbeitung unter CICS und für Batch-Verarbeitung direkt unterstützt.

Diese Unterstützung umfasst:

- Volle Integration der normalen CPG-Datei-Verarbeitungs-Logik.
- Automatische PSB-Verwaltung
- Interne Auffindung logischer Fehler
- Volle Kompatibilität der Quellenprogramme für VSE und Z/OS.

Direkte DL/I-Unterstützung

Im CPG wird der Zugriff zur Datenbank DL/I in den Rechenbestimmungen über die Befehle QSSA, USSA und DLI unterstützt. CPG baut dabei automatisch die internen DL/I-Calls auf, die vom Programmierer mit wenigen Operationen aufbereitet werden. CPG verwaltet automatisch PCBs und SSAs. Der Programmierer kann über interne CPG-Felder Return-Codes des DL/I abfragen. CPG setzt den Return-Code in das Feld vierstellige Alphafeld CPGDRC. Auch die Key-Feedback-Area kann über ein CPG-internes Feld verarbeitet werden. Das folgende Beispiel soll die Arbeitsweise verdeutlichen:

1. Einträge in der Files Division.

Es ist für alle Zugriffe zu DL/I eine File-Bestimmung pro Programm notwendig, die wie folgt aussieht:

```
- FILE MYPSBNM  DLI
```

Der Dateiname muss gleich dem verwendeten PSB-Namen sein. In der FCT muss die Zugriffsberechtigung für das verwendete Programm aufgenommen sein. Der Eintrag für die Einheit ist DLI.

2. Einträge in der Input Division:

```
     - FILE MYROOT DS
     -   1  8 MYCUNR                              (Segmentlänge)
     -   2  8 MYCUN7
     -   9 33 MYCUNM
```

Die Strukturen der Segmente werden als Datenstruktur in den Eingabebestimmungen beschrieben. Es kann sich auch um eine Überlagerung in der Data Division handeln. Wichtig ist, dass DL/I in der Länge des DL/I-Segmentes einliest. Daher sollte der Bereich immer in der maximalen Größe zur Verfügung gestellt werden.

3. Einträge in der Procedure Division:

3.1. DL/I Aufbau der SSA.

Segment Search Areas werden mit den Befehlen QSSA und USSA intern aufbereitet. Ein einfacher qualifizierter SSA-Aufbau sieht wie folgt aus:

```
- 'MYROOT  ' QSSA 'MYCUNR  ' KDNR 8 EQ
```

In F1 der QSSA-Operation wird der Name des Segmentes eingetragen. Er muss in Hochkommata eingeschlossen sein und darf maximal 8 Zeichen lang sein. In F2 wird, ebenfalls in Hochkommata, der Name des sensitiven Argumentes definiert. Im Ergebnisfeld wird der Feldname des programmabhängigen Schlüsselfeldes definiert. Das Feldlängenfeld wird benötigt, um die Anzahl der linksbündig verwendeten Stellen dieses Keyfeldes zu definieren. Es muss ein Vergleichsoperator als Bedingung (BD) angegeben werden; EQ,NE,GT,LT,GE und LE sind die möglichen Einträge.

Als Service-Funktion kann vor dem Operator ein 'O' oder 'A' angegeben werden. Über diesen Eintrag können boolsche Vergleichsoperationen über ein SSA gesetzt werden: 'A' bedeutet logisch und- , 'O' logisch oderverknüepft. Eine solche Verknüpfung zeigt das nächste Beispiel:

```
- 'MYROOT  ' QSSA 'MYCUN1  ' NAME 1 EQ
-            QSSA 'MYCUCY  ' ORT  1 A GE
```

Der Zugriff verläuft mit einem SSA auf das Segment MYROOT, wenn der Name des Kunden in der ersten Stelle gleich Name und der Ort des Kunden größer oder gleich dem eingegebenen Ort ist.

Unqualifiziertes SSA.

```
- 'MYROOT  ' QSSA 'MYCUNR  ' KDNR 5 EQ
- 'MYCADR  ' USSA
```

Bei einem unqualifizierten SSA-Aufruf wird in F1 der Segmentname wie beim qualifizierten SSA eingetragen. Die Operation heisst USSA. Weitere Einträge sind nicht notwendig.

3.2. DL/I Aufruf über die Operation DLI.

Der eigentliche Call zum DL/I wird über die Operation DLI erzeugt:

```
- GU DLI # MYROOT 2 10
```

Im F1 wird der DL/I-Call angegeben. Als Operation muss DLI eingetragen werden. F2 bleibt normalerweise frei, das heisst es muss ein Nummernzeichen (#) eingetragen werden. F2 kann aber auch einen zur Files Division unterschiedlichen PSB-Namen enthalten. Im Ergebnisfeld wird der Name des Ein-/Ausgabebereichs definiert. Im Beispiel ist dies der Name der Datenstruktur in der Eingabe. Die '2' im Feldlängenfeld gibt die Nummer des verwendeten PCBs an. Die Bezugszahl 10 wird gesetzt, wenn von DL/I ein Return-Code ungleich ' ' , 'GA' oder 'GK' gesetzt wird.

Es kann im Ergebnisfeld auch der Name eines Feldes definiert werden, das zwar in der Data Division definiert wurde, aber keine Datenstruktur darstellt. Zugriffe zum DL/I , die keine Daten übertragen, brauchen aus Kompatibilitätsgründen ebenfalls einen Eintrag im Ergebnisfeld, der dann beispielsweise ein Dummyfeld der Länge 1 sein kann.

Dazu als Beispiel ein Termination-Call:

```
- TERM DLI # DUMMY 1 10
```

Es wird auf die im Zugriff befindliche DB ein Termination-Call durchgeführt.

Beachte:

Beim Termination-Call wird vom DLI ein Syncpoint gesetzt. Daraus folgt für die Programmierung im CPG, dass wie bei der CPG-Operation SYNCP beschrieben, vor einem Termination-Call alle im Programm verwendeten Dateien mit RANDOM freigegeben werden müssen.

Abfrage Return-Code und KFBA.

```
     - INPUT DIVISION
     -   FIELD CPGSMN DD.              * Im Data Dictionary beschrieben
           1   8  SEGNAM
           9  10  SEGLEV
       BIN 11 14 0 KFBALE
           15 18  SENSGS
           19 146 KFBA
     - PROCEDURE DIVISION.
     -   IF CPGDRC > '  '.             * DL/I Return-Code
     -     SELCT CPGSMN
     -   END
     -
```

Die Key-Feedback-Area wird bei jedem DL/I-Aufruf mit den aktuellen Werten gefüllt. Der Aufbau dieser Area ist oben beschrieben. CPG füllt das interne Feld CPGDRC linksbündig mit dem zweistelligen DL/I-Return-Code.

Variable Einträge bei DL/I-Operationen 2335

Die Einträge der DL/I-Operationen können variabel gehandhabt werden.

Dazu muss im Programm eine 40-stellige Datenstruktur mit Namen CPGDLV zur Verfügung gestellt werden, deren einzelne Subfelder die alternativen Werte enthalten. Sind zur Ausführungszeit des Befehls die Subfelder der Datenstruktur ungleich blank bzw. Null, so wird ihr Wert statt des Wertes im DLI-, QSSA- oder USSA-Statement angenommen.

Als Service muss für diese Verarbeitungsart ein V oder VARiabel angegeben werden.

Die Datenstruktur muss dann folgenden Aufbau haben (eine analoge Redefinition von CPGDLV in der Data Division führt zum gleichen Ergebnis):

```
- INPUT DIVISION.
-   FILE CPGDLV DS DD.            * im Data Dictionary beschrieben
           1  8   SEGNAM.         * Segmentname
           9 16   SENKEY.         * sensitives Feld
      PAC 17 18 0 KEYLEN.         * Schlüssellänge
          19 20   COMPAR.         * Vergleichsoperator
          21 24   EXTENS.         * Erweiterung des Operators
          25 25   ANDOR.          * AND-/OR-Verknüpfung
          27 30   FUNCT.          * DLI-Call
          31 38   PSBNAM.         * PSB-Name
      PAC 39 40 0 PCBNUM.         * PCB-Nummer
```

Beispiel: Der Befehl

```
- 'ARTIROOT'  QSSA  'ARKO1   ' KEY  7  GE
```

kann somit auch wie folgt dargestellt werden:

```
-  SEGNAM = 'ARTIROOT'.
-  SENKEY = 'ARKO1   '.
-  COMPAR = 'GE'.
-  KEYLEN = 7.
- 'XXX     '  QSSA  'YYY     ' KEY  1  VARIABEL  EQ
```

Sind Subfelder der Datenstruktur CPGDLV nicht gefüllt, so wird der Wert des QSSA-Statements angenommen.

Datenview - Verarbeitung 2340

Datenview - Definition 2341

Eine Datenview ist eine Zusammenstellung von Daten für eine bestimmte Aufgabenstellung.

Eine View kann beliebig aus Feldern und Sätzen verschiedener Dateien zusammengesetzt sein. Die Logik einer View unterscheidet sich von anderen Speicherungsformen dadurch, dass grundsätzlich jedes ihrer Elemente als Schlüsselfeld dienen kann.

Realisierung einer Datenview 2342

In relationalen Datenbanken wird eine Datenview online erstellt und bleibt für die Zeit ihrer Verarbeitung im Hauptspeicher.

Um diese Verarbeitungsform zu simulieren, kann folgender Weg gewählt werden: Mit einem CPG2- oder CPG3..Query-Programm wird eine Datenview erstellt und auf der Datei CPGWKV abgestellt. Die genaue Vorgehensweise ist in Kapitel 7500 beschrieben.

Die View wird erst dann aus der Datei in den Hauptspeicher geladen, wenn sie in einem Programm angesprochen wird. Die geladene View bleibt dann bis zum folgenden Shut-Down des TP-Monitors im Hauptspeicher.

Die erstellte View hat die Form einer Tabelle mit der Besonderheit, dass jede Spalte der Tabelle als Schlüsselbegriff dienen kann.

Verarbeitung von Datenviews 2343

Zur Verarbeitung von Datenviews mit CPG muss beachtet werden:

- Für die View muss eine FILE-Bestimmung angelegt werden. Als Einheit muss TABLE angegeben werden, als Satzlänge die Summe der Felder der Tabelle und als Schlüssellänge das Maximum der Längen der möglichen Schlüsselfelder.

- Eine Datenview wird ähnlich wie eine Datei verarbeitet; alle Felder, die aus der Tabelle eingelesen werden sollen (insbesondere alle Schlüsselfelder), müssen in den Eingabebestimmungen unter dem Namen der Tabelle eingelesen werden.

- Gelesen wird die View mit dem Befehl FIND. Faktor 1 enthält das Schlüsselelement, Faktor 2 den Namen der View (bis zu vier Stellen). Eine Bezugszahl für den Vergleich auf Gleichheit zeigt an, ob das Suchargument aus Faktor 1 in der View gefunden wurde.

- Soll während der Verarbeitung wieder am Anfang der Tabelle aufgesetzt werden, so wird die sequentielle Verarbeitung der View mit der Operation RANDOM beendet. (Wurde ein mit FIND gesuchtes Element nicht gefunden, wird automatisch am Tabellenanfang wieder aufgesetzt.)

Beispiel:

Eine View wird aus Daten der Artikel- und Kundenstammdatei erstellt.

Beide Dateien haben nur jeweils einen Schlüssel: Die Artikelnummer bzw. Kundennummer. Diese Schlüssel werden im folgenden als 'primary key(s)' bezeichnet.

Mit der erstellten View können die Daten nicht nur nach den primary keys, sondern auch nach beliebig vielen anderen, im folgenden als 'secundary keys' bezeichneten Schlüsseln durchsucht werden. Beispiele für solche secundary keys sind Postleitzahl, erste Stelle der Postleitzahl oder Vertreternummer.

Beim Erstellen der View ist nun abzuwägen, wie viele 'Spalten' die generierte Tabelle haben soll. Insbesondere sollte zur sinnvollen Nutzung der View-Verarbeitung darauf geachtet werden, dass die View die primary keys der Dateien enthält, aus deren Daten sie sich zusammensetzt. Somit ist sichergestellt, dass nach einem erfolgreichen FIND auf sämtliche Daten (mit CHAIN) zugegriffen werden kann, die mit der View in Beziehung stehen. Vgl. auch Beispiel 17, Kapitel 8.

DC-Funktionen 2350

Hauptspeicher-Management 2351

DC-Programme bestehen im Gegensatz zu Batch-Programmen immer aus mehreren Teilen. Die erste Unterteilung erfolgt in einen Instruktionsteil und einem Datenteil, für die jeweils unterschiedliche Regeln gelten.

1) Instruktionen

Der Instruktionsteil ist so aufgebaut, dass er von mehreren Benutzern gleichzeitig verwendet werden kann. Dazu muss das Programm quasi-reentrant geschrieben sein, dass heisst zwischen zwei Macro-Instruktionen, die die Steuerung an den übergeordneten TP-Monitor übergeben, müssen die Programme so beschaffen sein, dass der Urzustand für den folgenden Benutzer wieder hergestellt wird; das heisst, es dürfen keinerlei Daten, Schalter oder Weichen im Programmtext abgespeichert werden. CPG Programme erfüllen diese Anforderung.

Instruktionsteile von Programmen werden von fast allen bekannten TP-Monitoren beim ersten Aufruf in den Speicher geladen und bleiben dort bis zum Abbruch der Online-Verarbeitung (Shut Down) liegen, wenn nicht vom System festgestellt wird, dass der vorhandene Speicher nicht ausreicht (Short on Storage Bedingung).

2) Daten

Bei den Daten unterscheiden wir je nach Zuordnung verschiedene Klassen.

2A) Programmabhängige Daten

Alle programmabhängigen Daten werden in einer Transaction Work Area gespeichert, die beim Aufruf eines Programms einem Bildschirm zugeordnet wird und solange reserviert wird, bis das Programm entweder durch die Ausführung der letzten Rechenbestimmung oder durch Betätigung der Löschtaste der Bildschirmtastatur normal oder nach einem Fehler abnormal beendet wird.

Die Transaction Work Area (TWA) wird dem Bildschirm temporär für die Dauer der Arbeit zugeordnet und kann damit mehrfach im Speicher sein, und zwar für jeden Bildschirm, der das Programm aufruft, einmal.

2B) Ein-/Ausgabe-Daten

Bei der Ein- oder Ausgabe von Daten wird unmittelbar vor der Ausführung der Ein-/Ausgabe-Operation ein Hauptspeicher-Segment angefordert, in das bei Ausgabe die Daten aus der Transaction Work Area übertragen werden und aus dem nach Ausführung der Eingabe-Instruktion die Daten in die Transaction Work Area übertragen werden. Danach wird dieser Bereich in der Regel sofort freigegeben, so dass Ein-/Ausgabebereiche den Hauptspeicher nur für extrem kurze Zeiten belasten.

Bei einigen Operationen müssen jedoch diese Bereiche bis zum Ende des Verarbeitungsvorgangs erhalten bleiben. Diese Ausnahmen sind:

Sequentielles Lesen von Platten	Der Bereich bleibt bis zum Ende der sequentiellen Verarbeitung (RNDOM oder Programmende) erhalten.
CHAIN für Update	Der Bereich bleibt bis zum Zurückschreiben des Satzes erhalten bzw. bis zum RNDOM für diese Datei.
Lichtstiftunterbrechung	Der Bereich bleibt bis zur folgenden Ein-/Ausgabe-Operation für ein Terminal erhalten.

2C) Programmunabhängige Daten

Schließlich können CPG2-Programme auch Daten untereinander austauschen. Die Daten müssen dazu in programmunabhängigen Bereichen zwischengespeichert werden. Die verschiedenen Verfahren hierzu sind im Abschnitt Zwischenspeicherung von Daten näher beschrieben.

SQL/DS 2360

CPG unterstützt SQL bzw. DB2 in der Produktstufe CPG3. Hierzu ist ein separates Installationsband erforderlich.

Die Programmierung der SQL-Befehle entspricht der normalen SQL-Syntax.

Im folgenden wird ein Miniprogramm aufgelistet,mit dem ein Zugriff auf die SQL-Tabelle KUNDEN durchgeführt wird. Es werden die Spalten KKDNR und KFIRMA gelesen und am Bildschirm angezeigt.

```
    - OPTIONS ROOT PHASE TST026 TITEL SQL-Testprogramm COMMANDLEVEL.
    - -D.
    -   SQL  BEGIN DECLARE SECTION
    -          USER    8
    -          PASSW   8
    -          KDNR    5 0
    -          FIRMA  30
    -   SQL  END DECLARE SECTION
    - -C.
    -   USER = 'SQLDBA  '
    -   PASSW = 'SQLDBAPW'
    -   SQL  CONNECT :USER IDENTIFIED BY :PASSW
    -   SQL  DECLARE C2 CURSOR FOR
    -   SQL  SELECT KKDNR,KFIRMA
    -   SQL  FROM KUNDENA
    -   SQL  WHERE KKDNR  < 3000
    -   SQL  OPEN C2
    -   DO LOOP
    -      SQL  FETCH C2
    -      SQL  INTO  :KDNR,:FIRMA
    -      IF SQCODE = 0
    -         MAPD ANZEIGE.                * von KDNR und FIRMA
    -      ELSE
    -         MAPD FEHLER.                 * Anzeige Return-Code
    -      ENDIF
    -      ON PF1  BREAK
    -   ENDDO
    -   SQL  CLOSE C2
    -   SQL  COMMIT WORK
 /*
```

Programmierhilfen 2400

Entscheidungstabellen 2410

CPG2 ermöglicht die direkte Verarbeitung von Entscheidungstabellen in den Rechenbestimmungen.

Durch die Operation BEGDT wird von der CPG2-Syntax auf Entscheidungstabellen-Logik und starre, spaltengebundene Schreibweise umgeschaltet.

Eine Entscheidungstabelle wird nicht durch CPG2-Statements unterbrochen und mit der Operation ENDDT abgeschlossen.

Eine Entscheidungstabelle besteht im oberen Teil aus Bedingungen, im unteren Teil aus Aktionen. Alle Bedingungen einer Entscheidungstabelle müssen unmittelbar aufeinander folgen, alle Aktionen ebenfalls.

Der Operationscode '-----' von Spalte 28 bis 32 gibt an, dass hier die Bedingungen enden und die Aktionen beginnen.

Die Spalten 43 bis 74 enthalten die Verknüpfungsleiste. Mögliche Eintragungen sind im Bedingungsteil je verfügbare Spalte:

Y	für 'YES' oder Bedingung erfüllt,
N	für 'NO' oder Bedingung nicht erfüllt,
BLANK	für: Bedingung ist nicht ausschlaggebend.

Mögliche Eintragungen für Aktionen sind ebenfalls je verfügbare Spalte:

Blank	Aktion wird nicht ausgeführt,
Nicht Blank	Aktion wird dann ausgeführt, wenn alle senkrecht in der gleichen Spalte darüber stehenden Bedingungen erfüllt,bzw. je nach Eintragung (N) nicht erfüllt sind.

Formularbeschreibung.

Allgemeine Eintragungen:

Spalte 6 'C' muss eingetragen werden.

Spalte 7-17 bleibt frei

Spalte 43-74 Verknüpfungsleiste (siehe oben).

Bedingungen.

Spalte 18-27 enthält einen gültigen Feldnamen, den Namen eines Feldgruppen-Elementes (FG,I) oder eine Konstante.

Spalte 33-42 enthält einen gültigen Feldnamen, den Namen eines Feldgruppen-Elementes (FG,I) oder eine Konstante.

Spalte 28-32 Operationsschlüssel. Gültige Eintragungen sind:

> oder 'HIGH' größer als
Faktor 1 ist größer als Faktor 2.

< oder 'LOW' kleiner als
Faktor 1 ist kleiner als Faktor 2.

= oder 'EQUAL' gleich.
Faktor 1 ist gleich Faktor 2.

Achtung: Bei allen Vergleichsoperationen müssen Faktor 1 und 2 entweder beide numerisch oder beide alphanumerisch definiert sein.

Aktionen.

Spalte 18-27 bleibt frei.

Spalte 28-32 Operationsschlüssel. Gültige Eintragungen sind :

GOTO	Verzweigen nach.
EXSR	Unterprogramm ausführen
EXCPT	Ausgabe

Aus einer Entscheidungstabelle darf jedoch keine zweite Entscheidungstabelle mit EXSR aufgerufen werden.

Beispiel.

Eine Kunden-Datei wird sequentiell verarbeitet, dabei sollen auf einem Bildschirm alle Kunden angezeigt werden, deren Umsatz größer als 10000 Euro und deren Soll-Saldo größer als der Umsatz ist, oder deren Umsatz größer als 10000 Euro und deren Saldo größer als ihr Kreditlimit ist. Alle anderen Kunden werden nicht angezeigt.

```
 - WEITER.
 -   READ BILD
 - LESEN.
 -   READ KUNDEN

 - LABEL BEGDT
 C           UMSATZ    >    10.000    YY
 C           SALDO     >    UMSATZ    Y
 C           SALDO     >    KREDIT     Y
 C           SALDO     >    10.000      Y
 C                     -----
 C                     EXSR PRÜF       X
 C                     EXCPT          XXX
 C                     GOTO WEITER    XXX
 C                     GOTO LESEN        X
 - ENDDT
```

Feldaufbereitung 2420

CPG bietet die Möglichkeit, Felder über die Output Division aufzubereiten. Mit der Operation MOVE können nur bis zu 8 Bytes als Konstante in ein Feld übertragen werden, mit der Operation '=' 24 Stellen.

Mit der Operation EDIT kann jedes alphanumerische Feld in voller Länge wie ein Ausgabesatz aufbereitet werden. Das Ergebnisfeld der Operation enthält den Namen eines Feldes, dessen Aufbereitung in der Output Division beschrieben ist (vgl. Operationscode EDIT).

Beispiel:

```
DATA DIVISION
  ZEILE  70
PROCEDURE DIVISION
  EDIT ZEILE
OUTPUT DIVISION
  FIELD ZEILE
     KDNR  7
     NAME  33
     ORT  55
     UMSATZ  70 ' .   . 0 ,  -'.     * aufbereitet mit Schablone
```

Umgekehrt können mit der Operation SELECT einzelne Felder aus dem aufbereiteten Feld wieder herausgelöst werden. In der SELCT-Operation wird der Name des Feldes angegeben, dessen Aufbau in der Input Division beschrieben sein muss; damit ist in der Input Division auch beschrieben, welche Teile des Feldes in welche anderen Felder übernommen werden.

Beispiel:

```
INPUT DIVISION.
   FIELD ZEILE
      1  7 KDNR
      8 33 NAME
      38 55 ORT
PROCEDURE DIVISION.
   SELECT ZEILE
```

Nach Ausführung der Operation enthält das Feld 'KDNR' die Bytes 1 bis 7, das Feld 'NAME' die Bytes 8 bis 33 und das Feld 'ORT' die Bytes 38 bis 55 des Feldes 'ZEILE'.

Feldaufbereitung mit TYPE 2425

Um verschiedene Strukturen aus gleichen Feldern einlesen und in gleiche Felder ausgeben zu können, kann ein Typ mit dem Schlüsselwort TYPE angegeben werden.

Beispiel für die Input Division (Output analog):

```
    - -I.  FIELD CPGCOM  TYPE  PROG1
    -         PACKED  1  4 0  KDNR
```

```
-                5 34    FIRMA
-      FIELD CPGCOM  TYPE  PROG2
-                1 20    PARAM

- -C.
-      SELECT CPGCOM TYPE PROG2
```

Im Beispiel arbeitet ein CPG-Programm über Programmverbindungen mit mehreren anderen Programmen zusammen. Der Datenaustausch erfolgt über die Common Area. Der SELECT-Befehl aus dem Beispiel oben bezieht sich auf die Common Area des Programms 2, die einen anderen Aufbau hat als beispielsweise der Verständigungsbereich zum Programm 1.

Beachte:

Im Programmcode in Input- und Output Division müssen die verschiedenen Typen einer Feldaufbereitung zusammenhängend (en bloc) codiert werden.

Strukturierte Programmierung 2450

CPG2 bietet zur Strukturierten Programmierung folgende Operationen an:

- BREAK (beende eine DO-Schleife bzw. alle DO-Schleifen)
- CAS (vergleiche und verzweige in eine Subroutine-Gruppe)
- CONTINUE (unterbreche einen Schleifendurchlauf)
- DO (führe aus)
- DO UNTIL (führe aus..bis)
- DO WHILE (führe aus..während)
- ELSE (sonst..führe aus)
- END (end)
- ENDDO (DO Ende)
- END-EVALUATE
- ENDIF (IF Ende)
- EVALUATE (genau eine von mehreren Alternativen ausführen)
- IF (wenn..dann)
- WHEN (wenn..dann in einer EVALUATE-Statementgruppe)

Die strukturierenden Operationen DO, IF, EVALUATE und CAS bilden jeweils den Anfang einer Statement-Gruppe, die mit einem END-Befehl abgeschlossen wird.

Als vergleichende Operatoren in DO-, IF- und WHEN-Abfragen können eingesetzt werden:

Operator		Bedeutung
>	GT	Faktor 1 ist größer als Faktor 2
<	LT	Faktor 1 ist kleiner als Faktor 2
=	EQ	Faktor 1 ist gleich Faktor 2
>< <>	NE	Faktor 1 ist ungleich Faktor 2
>=	GE	Faktor 1 ist größer oder gleich Faktor 2
<=	LE	Faktor 1 ist kleiner oder gleich Faktor 2

Beispiel:

```
DO WHILE ERROR = '  '
  IF WERT1 > WERT2
     :
  END
END
```

Enthält eine DO-Gruppe eine andere vollständige DO- oder IF-Gruppe, so spricht man von einer geschachtelten DO-Gruppe. Im Beispiel ist die Schachteltiefe 2, maximal 40 Ebenen sind von CPG unterstützt.

Ein CPG-Programm kann insgesamt höchstens 999 DO- und IF-Befehle enthalten.

AND-Verknüpfung 2453

Durch logische Verknüpfung sollen mehrere Bedingungen in einem Programmschritt geprüft werden. Die AND-Verknüpfung ist für die Operationen IF, DO und WHEN unterstützt.

(Siehe dazu auch Kapitel Operationen - IF-Anweisung und Beispiel 26)

BREAK-Operation 2455

Die Operation BREAK beendet die aktuelle DO-, DO UNTIL- oder DO WHILE-Schleife und verzweigt hinter das zugehörige END-Statement. BREAK ALL verzweigt hinter das END der äußersten DO-Schleife.

CONTINUE-Operation 2457

Die Operation CONTINUE unterbricht die Verarbeitung der aktuellen DO-, DO UNTIL- oder DO WHILE-Schleife.

CONTINUE verzweigt vor das END-Statement der Schleife und somit wieder zurück zur Schleifenbedingung. Die Statements zwischen CONTINUE und END werden somit nicht ausgeführt, die Schleife wird aber entsprechend der programmierten Schleifenbedingung weiterhin durchlaufen.

DO-Operation 2460

Beispiel: DO FROM X TO Y WITH I

Die DO-Operation funktioniert in folgender Weise:

Beim DO wird der Anfangswert zuerst in den Index übertragen.

Der Index wird mit dem Begrenzungswert verglichen. Ist der Index grösser als der Begrenzungswert, verzweigt die Programmsteuerung zu der Rechenbestimmung, die der zugehörigen END-Bestimmung folgt.

Ist der Index kleiner oder gleich dem Begrenzungswert, werden die Rebestimmungen zwischen dem DO und dem zugehörigen END ausgeführt.

Im END-Befehl der DO-Gruppe wird der Erhöhungsfaktor der Schleife zum Indexfeld addiert. (Der Erhöhungsfaktor ist in der Regel gleich 1,kann aber auch hinter dem ENDDO-Befehl als positive Zahl angegeben werden.) Anschließend verzweigt die Programmsteuerung zurück zum zugehörigen DO und vergleicht erneut Indexwert und Begrenzungswert.

Um eine DO-Schleife zu beenden kann innerhalb der Schleife der Begrenzungswert verändert werden. Eine Indexänderung ist nicht möglich. Weitere Möglichkeiten der Schleifenunterbrechung bieten die Operationen BREAK und CONTINUE.

Nach Beendigung der Schleife ist der Indexwert größer als der Begrenzungswert. Bei einer Schleife, die von 1 bis 12 läuft (mit der Erhöhung von 1 pro Schleifendurchlauf) ist der Index nach der Ausführung gleich 13.

Eine DO-Statement-Gruppe endet mit dem Befehl END oder ENDDO. Wird ein Erhöhungswert für den Schleifenindex angegeben, so muss die Operation ENDDO gewählt werden.

DO UNTIL - Operation 2461

Die Operation DO UNTIL funktioniert in folgender Weise:

Die Statements der DO-Gruppe werden solange ausgeführt, bis die angegebene Bedingung erfüllt ist. Nach jedem Durchlauf durch die DO-Gruppe wird die Bedingung erneut geprüft. Ist die Bedingung erfüllt, wird die Schleife nicht durchlaufen, sondern zum Statement verzweigt, das der zugehörigen END-Bestimmung unmittelbar folgt.

Die Bedingung für den Schleifendurchlauf kann sich aus mehreren logisch verknüpften Bedingungen zusammensetzen. Es stehen dafür die Operatoren AND und OR zur Verfügung.

Durch die Servicefunktion '1' im Anschluß an ein DO UNTIL-Statement wird erreicht, dass die Schleife beim ersten Durchlauf ohne Prüfung der Schleifenbedingung (also grundsätzlich ein Mal) durchlaufen wird.

DO WHILE - Operation 2462

Die Operation DO WHILE funktioniert in folgender Weise:

Die Statements der DO-Gruppe werden solange ausgeführt, wie die angegebene Bedingung erfüllt ist. Nach jedem Durchlauf durch die DO-Gruppe wird die Bedingung erneut geprüft. Ist die Bedingung erfüllt, wird die Schleife durchlaufen. Ansonsten wird zum Statement verzweigt, das der zugehörigen END-Bestimmung unmittelbar folgt.

Die Bedingung für den Schleifendurchlauf kann sich aus mehreren logisch verknüpften Bedingungen zusammensetzen. Es stehen dafür die Operatoren AND und OR zur Verfügung.

ELSE-Operation 2463

Eine ELSE -Operation gibt den Anfang derjenigen Rechenbestimmungen an, die ausgeführt werden, wenn die Prüfungen der zugehörigen IF-Operation nicht zutreffen.

Die Operation ELSE ist im CPG immer ein separates Statement.

END-Operation 2464

Die Operation END muss jede DO- und IF-Gruppe abschließen.

Soll in einer DO-Schleife der Erhöhungswert ungleich 1 sein, so kann dieser als Faktor 2 der Operation ENDDO als einstellige Konstante angegeben werden.

Zur Dokumentation kann statt END auch ENDIF oder ENDDO programmiert werden.

EVALUATE-Operation 2465

Die EVALUATE-Operation wird eingesetzt, wenn (höchstens) eine von mehreren Alternativen ausgeführt werden soll.

Als Operationsbezeichnung muss "EVALUATE" angegeben werden. Die Alternativen werden mit "WHEN" in den Folgezeilen gekennzeichnet. Die Faktoren für die Bedingungen können numerische oder alphanumerische Felder, Feldnamen oder Feldgruppenelemente enthalten. Ist eine Bedingung erfüllt, werden die nachfolgenden Anweisungen bearbeitet. Danach ist die Evaluate-Anweisung beendet und das Programm wird hinter dem "END-EVALUATE" fortgesetzt.

Die Bedingung "WHEN OTHER" ist erfüllt, wenn keine der vorhergehenden WHEN-Anweisungen zutreffend war. Die zugehörigen Anweisungen werden in diesem Fall ausgeführt und das EVALUATE ist beendet.

IF-Operation 2466

Die IF-Operation funktioniert in folgender Weise:

Die beiden Faktoren des Vergleichs müssen vom gleichen Typ sein. Sie enthalten entweder eine alphanumerische oder numerische Konstante, ein Feld oder ein Feldgruppenelement.

Besteht die Beziehung zwischen Faktor 1 und Faktor 2 nicht, verzweigt die Programmsteuerung zu der Recheninstruktion, die der zugehörigen END-bzw. ELSE-Anweisung unmittelbar folgt.

Eine END-Bestimmung muss eingetragen werden, um eine IF-Operation ab-

zuschließen. Folgt einer IF-Bestimmung eine ELSE-Bestimmung, muss die END-Anweisung nach der ELSE-Bestimmung und nicht nach der IF-Bestimmung eingetragen werden.

IF CONDITION 2467

IF CONDITION funktioniert in folgender Weise:

Es werden nur Bezugszahlen und andere Schalter abgefragt. Die abgefragte Bedingung kann aus bis zu drei Schaltern bestehen, die logisch und-verknüpft sind.

Ist die Bedingung erfüllt, so werden die Statements ausgeführt, die zwischen dem IF und dem nächsten END oder ELSE codiert sind. Ist die Bedingung nicht erfüllt, so wird hinter das nächste END oder ELSE verzweigt.

OR-Verknüpfung 2468

Durch boolesche Verknüpfung sollen mehrere Bedingungen in einem Programmschritt geprüft werden. Die logische Oder-Verknüpfung ist für Operationen IF, DO und WHEN unterstützt.

(Siehe auch Kapitel Operationen - IF-Anweisung und Beispiel 26)

WHEN 2470

WHEN funktioniert in folgender Weise:

Die WHEN-Operation gibt eine Bedinung an. Ist diese erfüllt, werden alle nachfolgenden Anweisungen bis zum nächsten WHEN oder bis zum END-EVALUATE ausgeführt.

WHEN OTHER 2471

WHEN OTHER ist der 'ELSE'-Zweig der mehrfachen Alternative (EVALUATE):

Wenn keine der vorhergehenden WHEN-Bedingungen erfüllt war, werden die Befehle der WHEN-OTHER-Gruppe (also bis zum END-EVALUATE) ausgeführt.

WHEN OTHER ist (optional) die letzte Bedingungsabfrage in jeder EVALUATE-Gruppe.

Boolesche Verknüpfung von IF-und WHEN-Operationen 2475

IF-und WHEN-Ooperationen können mit booleschen Operatoren OR und AND logisch verknüpft werden.

1. Oder-Verknüpfung mit OR bei der IF-Operation

Die Statements bis zum nächsten END oder ELSE werden ausgeführt, wenn mindestens eine der IF-Bedingungen erfüllt ist. Ist keine der Bedingungen erfüllt, wird hinter das nächste END oder ELSE verzweigt.

2. Oder-Verknüpfung mit OR bei der WHEN-Operation

Die Statements bis zur nächsten WHEN-Anweisung werden durchgeführt, wenn eine oder beide WHEN-Bedingungen erfüllt sind. Ist keine Bedingung erfüllt, wird das Programm bei der nächsten WHEN-Operation fortgesetzt.

3. Oder-Verknüpfung mit OR bei der DO UNTIL / DO WHILE-Operation

Die Statements bis zum nächsten END werden ausgeführt, wenn mindestens eine der DO-Bedingungen erfüllt ist. Ist keine der Bedingungen erfüllt, wird hinter das nächste END verzweigt.

4. Und-Verknüpfung mit AND bei der IF-Operation

Die Statements bis zum nächsten END oder ELSE werden ausgeführt, wenn alle IF-Bedingungen erfüllt sind. Ist nur eine der Bedingungen nicht erfüllt, wird hinter das nächste END oder ELSE verzweigt.

5. Und-Verknüpfung mit AND bei der WHEN-Operation

Die Statements bis zur nächsten WHEN-Anweisung werden durchgeführt, wenn beide WHEN-Bedingungen erfüllt sind. Ist eine der beiden oder sind beide nicht erfüllt, wird das Programm bei der nächsten WHEN-Operation fortgesetzt.

6. Und-Verknüpfung mit AND bei der DO UNTIL / DO WHILE Operation

Die Statements bis zum nächsten END werden ausgeführt, wenn alle Do-Bedingungen erfüllt sind. Ist nur eine der Bedingungen nicht erfüllt, wird hinter das nächste END verzweigt.

OR und AND können beliebig gemischt verwendet werden. Dabei gilt die Regel, dass AND stärker bindet als OR (analog der Regel 'Punktrechnung geht vor Strichrechnung').

Zu der verknüpften IF-Bedingungen gehört immer nur ein END (IF), die Operation WHEN wird nicht mit END beendet.

Daten-Strukturen 2477

CPG erlaubt, einen Bereich im Speicher zu bestimmen sowie das Anlegen von Feldern - genannt Subfelder - innerhalb dieses Bereiches. Dieser Bereich im Speicher wird Datenstruktur genannt.Eine Datenstruktur kann benutzt werden, um

. denselben internen Bereich mehrfach unter Verwendung verschiedener Datenformate zu beschreiben,

. mit einem Feld zu rechnen und seine Inhalte zu verändern,

. ein Feld in Subfelder zu teilen, ohne MOVE oder MOVEL-Befehle zu benutzen,

. eine Datenstruktur und seine Subfelder auf die gleiche Art zu beschreiben, wie ein Satz definiert wird,

. nicht zusammenhängende Daten in zusammenhängende interne Speicherbereiche zu gruppieren.

Datenstrukturanweisungen werden in der Input Division wie folgt beschrieben:

```
- FILE  Name  DS  (Länge)
```

Folgende Regeln sind beim Spezifizieren von Datenstrukturanweisungen zu berücksichtigen:

. Der Datenstrukturname muss ein gültiger, maximal 30 Stellen langer symbolischer Name sein. Er kann überall dort angesprochen werden, wo ein alphanumerisches Feld erlaubt ist.

. Alle Eintragungen für eine Datenstruktur und ihre Subfelder müssen zusammen erscheinen; sie können nicht mit Eintragungen für andere Datenstrukturen gemischt werden.

. Subfelder wirken definierend für die Datenstruktur; deshalb dürfen Subfelder nicht gleichzeitig in verschiedenen Datenstrukturen liegen

. Die Länge einer Datenstruktur kann wie folgt sein:
  - Die Länge, angegeben in den Eingabefeldbestimmungen, wenn der Datenstrukturname ein Eingabefeldname ist.
  - Die höchste Bis-Stelle eines Subfeldes innerhalb einer Datenstruktur, wenn der Datenstrukturname kein Eingabefeld ist.
  - Die wahlweise anzugebende Länge in der oben beschriebenen Anweisung in der Input Division

. Die Länge der Datenstruktur wird bestimmt durch die erste Anweisung im Programm, die eine Länge der soeben beschriebenen Arten definiert. Nachfolgende anderslautende Längenangaben sind ungültig.

. Eine Datenstruktur und ein Subfeld einer Datenstruktur können nicht denselben Namen haben.

. Wird eine SELCT-Operation für eine Datenstruktur benutzt, so müssen die SELCT-Eingabe-Bestimmungen **vor** den Datenstruktur-Bestimmungen

liegen.

. Ist ein Feld in einer Datenstruktur definiert, so darf es nicht in der Data Division unterdefiniert werden.

Datenstruktursubfeld-Bestimmungen 2480

Die Subfelder einer programmbeschriebenen Datenstruktur müssen unmittelbar auf die Datenstrukturanweisung folgen, zu der sie gehören. Die Syntax ist die gleiche wie für andere Feld-Eingaben.

Verwendung einer Datenstruktur, um Subfelder innerhalb eines Feldes zu definieren:

```
- FILE INPUT
-   3 18 PARTNO
-  19 29 NAME
-  30 40 PATNO
-  41 61 DR
- FILE PARTNO DS.  1  4   MFG
-                  5 10   DRUG
-                 11 13   STRNTH
-            PAC  14 16 0 COUNT
```

Die Datenstruktursubfelder können über den Namen PARTNO oder über die Subfeldnamen MFG, DRUG, STRNTH oder COUNT angesprochen werden.

Verwendung einer Datenstruktur zur Zusammenstellung von Feldern:

```
- FILE INPUT
-    3 10  PARTNO
-   11 16 0 QTY
-   17 20  TYPE
-   21 21  CODE
-   22 25  LOCATN
- FILE PRTKEY DS.
-    1  4 LOCATN
-    5 12 PARTNO
-   13 16 TYPE
```

Wird eine Datenstruktur benutzt, um Felder zu gruppieren, können Felder von nicht benachbarten Stellen auf dem Eingabesatz zusammenliegend gruppiert werden. Auf diesen Bereich kann dann durch den Datenstrukturnamen und/oder den individuellen Subfeldnamen zugegriffen werden.

Zur Zeit können maximal 500 Eintragungen in die Datenstrukturtabelle aufgenommen werden. Eine Erweiterung ist bei Bedarf möglich.
Siehe Copy CPG*CDTB. (* = Releasesuffix).

TWA - Überlagerung 2485

Es können auch Überlagerungsfelder (Overlay-Felder) definiert werden. Ein Überlagerungsfeld ist ein Feld, das in weitere Felder unterteilt ist. Es wird in der Data Division wie eine Feldgruppe definiert, als Anzahl der Elemente wird aber '0' angegeben.

Überlagerungsfelder können numerisch oder alphanumerisch sein.

Auf jede Spezifikation eines Überlagerungsfeldes müssen die Definitionen der Felder folgen, in die der Bereich unterteilt werden soll. Diese werden als gewöhnliche Felder oder Feldgruppen definiert.

Ein Überlagerungsfeld kann auch weitere Überlagerungen enthalten.

Es ist darauf zu achten, dass genügend Felder definiert werden, um den gesamten für ein Überlagerungsfeld definierten Bereich zu füllen. Als Kontrolle kann das Blockdiagramm herangezogen werden. Um davon Gebrauch machen zu können, darf allerdings pro Zeile nur eine Felddefinition vorgenommen werden.

Beispiel:

```
- ANSCHR   0 * 60.   NAME  20.  ORT  20.  STR  20.
```

Die drei Felder NAME, ORT, STR können infolge der Überlagerung auch unter dem gemeinsamen Namen 'ANSCHR' angesprochen werden; beim Einlesen des Feldes ANSCHR werden auch die Felder NAME, ORT und STR gefüllt.

Bei numerischen Feldern ist bei Überlagerung zu beachten, dass diese intern in gepacktem Format gespeichert sind.

Lichtstift - Auswahl 2490

Beim Einlesen des Bildschirms ist es möglich, nur ein bestimmtes Feld gezielt auszuwählen. Diese Auswahl geschieht durch Antippen mit dem Lichtstift bzw. indem man den Cursor auf das auszuwählende Feld setzt und zur Auswahl die Taste 'Pos Ausw' (für Positions-Auswahl) drückt.

Nach dieser sogenannten Lichtstift-Auswahl ist im Programm

- der Schalter SP gesetzt
- das 6-stellige alphanumerische Feld CPGMFN mit dem Namen des ausgewählten Feldes gefüllt (bei Einsatz von QSF)
- das 3-stellige numerische Feld CPGMFI mit dem Index des ausgewählten Elementes gefüllt, falls das ausgewählte Feld eine Feldgruppe war (bei Einsatz von QSF)

Zu beachten ist bei der Lichtstiftauswahl folgendes :

Jedes lichtstiftauswählbare Feld muss im ersten Byte ein Leerzeichen (Blank) haben.

Es dürfen sich nicht gleichzeitig lichtstiftauswählbare Felder und Felder, bei denen durch das Attribut 'Modified Data Tag' gesetzt ist,

in einer MAP-Befinden. Dies würde zu Fehlern führen, da das 'Modified Data Tag' monitor-intern genauso gehandhabt wird wie die Lichtstiftauswahl.

Cursor - Auswahl 2495

Die Lichtstiftauswahl hat an Bedeutung verloren. Heute können solche Auswahlen durch einfache Cursorpositionierung erreicht werden.

Bei jeder Bildschirmeingabe werden die internen Felder CPGMFN und CPGMFI gefüllt.

CPGMFN enthält den Feldnamen, in dem der Cursor steht

CPGMFI enthält den Index des Feldgruppenelements, falls der Cursor in einer Feldgruppe steht.

Optimierung der TWA-Größe 2500

Standardmäßig wird vom CPG die TWA-Größe minimiert. Felder, die in der Input Division (implizit) definiert sind, werden nicht in die TWA übernommen, wenn sie weder in der Procedure Division noch in der Output Division nochmals angesprochen werden.

Die Optimierungsfunktion wird nicht aktiviert für Felder, die explizit in der Data Division vereinbart wurden.

Die Optimierungsfunktion wird auch nicht wirksam bei Datenstruktur-Subfeldern. Die nicht optimierten Felder werden intern als 'used' gekennzeichnet; Datenstruktur-Subfelder sind also grundsätzlich 'used'.

Die Optimierung kann aufgehoben werden mit dem Options-Parameter DEF für Define.

Regeln bei transaktionsorientierter Programmierung 2550

. Die Lösch-Taste oder eine andere Programmfunktion für das Programmende muss vom Programmierer abgefragt werden.

. Für den transaktionsorientierten Programmaufruf steht in erster Linie die Operation EXITT zur Verfügung.

. Die Bildschirmdaten werden mit der Operation MAP ins Programm übertragen.

. Die Daten der TWA (Transaction Work Area) stehen nach dem Task-Ende nicht mehr zur Verfügung. Die für den Programmablauf relevanten Daten müssen zwischengespeichert werden (z.B. in einer Temporary Storage Queue) und beim Start der nächsten Task wieder in das Programm

eingelesen werden. Es ist zu beachten, ob diese Zwischenspeicherbereiche zum Ende der Anwendung gelöscht werden sollen.

Am Taskende werden die VSAM-Strings freigegeben. Das ist zum Beispiel zu beachten, wenn eine VSAM-Datei sequentiell gelesen wird.

QSF - Quick Screen Facility 2570

QSF bietet die Möglichkeit, Bildschirm-Maps interaktiv zu erstellen und Zu warten. QSF greift dabei auf eine Schnittstelle zu, die es dem Be-Nutzer erlaubt, alle im Programm definierten Felder oder eine Auswahl daraus auf dem Bildschirm nach seinen Erfordernissen anzuordnen.

Aufbau und Veränderung der Maps erfolgt dabei interaktiv und setzt keine Programmumwandlung voraus, auch dann nicht,wenn sonstige im Programm definierte Felder in die Map neu aufgenommen werden.
Farben, Texte,Feldeigenschaften und Positionen können durch einfaches Positionieren des Cursors und entsprechende Eintragungen über das QSF verändert werden.

Das Programm enthält weder Ein- noch Ausgabe-Bestimmungen für den Bildschirm. CPG stellt dem QSF die erforderliche Schnittstelle über die Operationen MAP, MAPD, MAPI, MAPO und MAPP zur Verfügung.

QSF ist in CPG2 und CPG3 enthalten.

QSF ist beschrieben im Handbuch CPG2-Serviceprogramme.

QLF - Quick List Facility 2580

QLF ist ein Programmpaket, das ab dem Service-Level CPG4 im CPG-Umfang enthalten isrt. Es bietet die Möglichkeit, Listbilder interaktiv zu erstellen und zu warten. QLF greift dabei auf eine Schnittstelle zu, die es dem Benutzer erlaubt, alle im Programm definierten Felder oder eine beliebige Auswahl daraus nach seinen Erfordernissen zu einem Listbild anzuordnen.

Aufbau und Veränderung der Listen erfolgt dabei interaktiv und setzt keine Programmumwandlung voraus, auch dann nicht,wenn sonstige im Programm definierte Felder neu in das Listbild aufgenommen werden.

Das Programm enthält keine Ausgabebestimmungen mehr für den Drucker.

CPG stellt dem QLF die erforderliche Schnittstelle über die Operation LIST für die Online- und Batchprogrammierung zur Verfügung.

Voraussetzung für QLF ist die Installation des Textverarbeitungs-systems QTF.

QLF ist beschrieben im Handbuch QTF.

Data Dictionary in der Files Division 2605

Data Dictionary wird bei der Dateibeschreibung grundsätzlich angezogen, wenn Dateien im Source-Code nur unvollständig beschrieben sind. Dabei wird die Beschreibung der Satzart ' ' (2 Blank) ins Programm übernommen.

Wie in den anderen Divisions können hier die Schlüsselwörter DD (für Data Dictionary) und TYPE (für die Satzart) angegeben werden, um eine unterschiedliche Dateibeschreibung gezielt anzusprechen.

Anwendungsbeispiele:

- Unterscheidung nach Input-, Output- und Updatedateien im Batch
- Unterscheidung nach physischer Datei und HL1-Dataset

Codierung:
```
FILE KUNDEN  DD  TYPE  HD
FILE KUNDEN  DD  TYPE  HD INPUT
```

Data Dictionary in der Data Division 2610

1. DEFINE Struktur

Strukturen, die im Data Dictionary beschrieben sind, können mit DEFINE 'Strukturname' in die Data Division übernommen werden.

Soll eine Satzart der Struktur angezogen werden, so wird diese zweistellig an den DEFINE-Befehl angehängt (siehe Beispiel).

Optional kann für die Satzart wie in den übrigen Divisions das Schlüsselwort TYPE angegeben werden.

In diesen Fällen werden die Feldbeschreibungen des Data Dictionary zur Umwandlungszeit in die Data Division übernommen.

Beispiele:

```
- -D.  DEFINE KUNDEN
-      DEFINE CPGWRK 80
-      DEFINE CPGWRK TYPE 80
```

Achtung: Das Schlüsselwort DD ist in dieser Division im Gegensatz zu den anderen Divisions für die Data Dictionary-Verarbeitung nicht erforderlich.

Die Eintragung DD hinter einer DEFINE DATEI-Definition wird als Satzart DD interpretiert.

Zur 'Technik': Bei DEFINE DD wird ein Filler generiert, falls die Struktur lückenhaft ist. (Bei Dateien mit Directory Field Check).

Beispiel:

```
DEFINE DEMO
  F1              10   * Feld 1
    CPGFIL        10   * CPGFILLER
  F2              20   * Feld 2
  F3              15   * Feld 3
    CPGFIL        65   * CPGFILLER
  F4              20   * Feld5
```

2. DEFINE Multiple

Wenn das gleiche Feld in der Data Division mehrfach definiert wird, kommt es zu der Fehlermeldung ' ...doppelt definiert '. Arbeitet man in der Data Division mit DEFINE Struktur, dann tritt dieser Fall auf, wenn ein Feld in mehreren Strukturen liegt. Durch DEFINE Multiple ist es möglich, ein Feld mehrfach in der Data Division zu definieren.

Beachte:

Diese mehrfach definierten Felder werden intern auf Kommentar gesetzt, dürfen also nicht Teil einer Redefinition (Überlagerung) sein.

Beispiel:

```
DEFINE DATEI M.
DEFINE DATEI TYPE xx M.
DEFINE DATEI TYPE XX MULtiple
DEFINE HQTFC MULtiple
```

Beide Strukturen enthalten das achtstellige Alphafeld DOKUM. Ohne MUL würde die Kompilierung mit der Meldung 'Feldname doppelt definiert abbrechen.

3. Standardwerte bei fehlender Felddefinition

Wenn nur Feldnamen eingetragen werden, so wird die Felddefinition dem QDDS entnommen. Lange Feldnamen sind in Verbindung mit DDL in der Options-Karte unterstützt.

Data Dictionary in der Input Division 2620

In der Input Division werden Data Dictionary-Strukturen durch Anhängen des Schlüsselworts DD an die Satzbestimmung angezogen.

Data Dictionary ist für FILEs und FIELDs unterstützt.

Soll eine Satzart einer Struktur angezogen werden, so wird diese mit dem Schlüsselwort TYPE an den Befehl angehängt.

Beispiele:

```
- -I.  FILE  KUNDEN DD.
-      FILE  CPGWRK DD  TYPE  80
-      FILE  CPGKSD DD  TYPE  AA  LIST
```

Durch die Schlüsselworte 'LISt' oder 'TEXt' erreicht man, dass zu jedem Feld auch eine Kommentarzeile generiert wird, die den beschreibenden Text aus dem Data Dictionary anlistet.

HL1-Datenkanäle werden ebenfalls im Data Dictionary beschrieben. Hierbei ist zu unterscheiden, ob der Datenkanal optimiert werden soll oder nicht. Ist keine Optimierung gewünscht, so gibt man das Schlüsselwort HS an und im Anschluss die Einträge für Data Dictionary. Zum Optimieren des Kanals lässt man den Eintrag HS weg. CPG stellt sicher, dass das Steuerfeld CPGHIC in jedem Fall im Programm erhalten bleibt.

Bei Datenstrukturen wird vor den Einträgen für Data Dictionary das Schlüsselwort DS angegeben.

Bei Feldaufbereitungen kann das Schlüsselwort TYPE verschiedene Bedeutungen haben. TYPE kann für die Satzart des Data Dictionary stehen, aber auch für die Auswahl der Feld-Selektion. Aus Gründen der Eindeutigkeit kann für die Feldauswahl auch das Schlüsselwort SELECT-type angegeben werden.

Beispiele:

```
-      FILE  KANAL  HS  DD  TYPE 02
-      FILE  STRUKT DS  DD
-      FIELD CPGCOM DD TYPE XS  TYPE PROGRAMM-5
```

Data Dictionary in der Input Division mit Feldabfragen 2625

Die Eingabe kann bei Einsatz von Data Dictionary von der Prüfung zweier Zeichen des Eingabesatzes abhängig gemacht werden.

Beispiele:

```
- -I.  FILE KUNDEN  DD            01  1 C    0  2 C    1
-      FILE KUNDEN  DD TYPE 99    02  1 CHAR 9  2 CHAR 9
```

Die Syntaxregel: 1. die vollständigen Data Dictionary-Einträge
2. die Bezugszahl oder das Auslassungszeichen #
3. die Abfrage von einem oder zwei Zeichen

Data Dictionary in der Output Division 2630

Strukturen des Data Dictionary können auch in der Output Division angezogen werden. Es gilt grundsätzlich, dass direkt im Anschluss an FILE 'Strukturname' die Eintragungen folgen müssen. Wie in der Input Division wird ein DD für Data Dictionary und gegebenenfalls zusätzlich ein TYPE 'Satzart' für die Satzart angegeben.

Die übrigen möglichen Eintragungen für die Satzbestimmung in der Output Division folgen in der bekannten Reihenfolge, die weiter oben beschrieben ist.

Bei Feldaufbereitungen kann das Schlüsselwort TYPE verschiedene Bedeutungen haben. TYPE kann für die Satzart des Data Dictionary stehen, aber auch für die Auswahl der Feldaufbereitung. Aus Gründen der Eindeutigkeit kann für die Feldauswahl auch das Schlüsselwort EDIT-type angegeben werden.

Beispiele:

```
- -O. FILE KUNDEN DD
-     FILE KUNDEN DD ADD NAME-DES-EXCPT
-     FILE KUNDEN DD ADD ON 99
-     FILE KUNDEN DD     ON 01 AND NOT 02
-     FILE CPGWRK DD TYPE 80
-     FILE CPGWRK DD TYPE 80 ADD
-     FILE CPGWRK DD TYPE 80 ADD ADD
-     FILE CPGWRK DD TYPE 80     ON 01 AND NOT 02 UPDATE-CPGWRK

-     FIELD A  DD
```

Mit Y oder T gekennzeichnete Keyfelder werden bei DISK- und KSDS-Dateien bei der Ausgabe für Update als Kommentar-Statements generiert. Beim Hinzufügen werden auch die Schlüsselfelder als Ausgabefelder generiert.

Bei Feldaufbereitungen kann Data Dictionary eingesetzt werden. In diesem Fall wird auch der Edit-Code, der im Data Dictionary angegeben ist, berücksichtigt.

Referenzstrukturen bei Data Dictionary-Verarbeitung 2640

Analog zur Referenzdatei, wie sie im Data Dictionary angegeben werden kann (siehe Handbuch CPG2-Serviceprogramme), kann auch in der Input und der Output Division mit Referenzstrukturen gearbeitet werden.

Durch diese Verarbeitungsart wird nicht die Struktur angezogen, die mit FILE 'Datei' definiert wurde, sondern unter diesem Namen wird die Struktur angezogen, die mit REF 'Datei2' zusätzlich angegeben wurde.

Beispiel:

```
- -I. FILE KUNDEN DD REF TEST01
-     FILE        DD TYPE 09 REF TEST01
-  :
- -O. FILE KUNDEN DD REF TEST01             KDUPD
-     FILE KUNDEN DD REF TEST01 TYPE 01 ADD KDADD
-     FILE        DD REF TEST01 TYPE 02 ADD KDADD
```

Bei der Verarbeitung der Datei KUNDEN wird eine Struktur eingelesen, die aus den Strukturen TEST01 und TEST01, Satzart 09 zusammengesetzt ist. Beim Update wird die Struktur TEST01 ausgegeben, beim Hinzufügen wird eine Struktur ausgegeben, die sich aus Satzart 01 und Satzart 02 der Struktur TEST01 zusammensetzt.

Das Schlüsselwort REF kennzeichnet die nachfolgend genannte Struktur als Referenzstruktur. REF steht immer unmittelbar hinter dem Schlüsselwort DD oder hinter der Angabe einer Satzart mit TYPE. Unabhängig von der Position im Befehl bezieht sich eine Satzart immer auf die Referenzstruktur.

Die Kombinationen mit anderen Einträgen werden durch den Parameter REF nicht eingeschränkt.

Durch Weglassen des Dateinamens wird erreicht, dass unter einem Dateinamen mehrere Strukturen zusammengesetzt werden können. Ohne diese Möglichkeit würde jeweils eine vollständige Satzbestimmung generiert. Beim Wechsel der Satzbestimmung endet aber bei der Ausführung der Einlese- bzw. Ausgabevorgang.

Data Dictionary und Optimierungsfunktion des CPG 2670

Grundsätzlich ist im CPG die Optimierungsfunktion aktiv. Das bedeutet, dass alle in der Input Division definierten Felder, die im weiteren Programmverlauf nicht mehr benutzt werden, vom Compiler ignoriert und folglich auch nicht angezeigt werden.

Diese Optimierungsfunktion kann durch den Options-Parameter DEF aufgehoben werden. Der Parameter DEF definiert explizit alle in der Input Division beschriebenen Felder.

Beispiel: Der Options-Parameter DEF muss dann angegeben werden, wenn Daten von einer Datei gelesen werden und ohne Verarbeitung mit einer QSF-Map am Bildschirm ausgegeben werden.

In diesem Fall werden die Felder in der Input Division beschrieben, aber weder in der Procedure Division noch in der Output Division nochmals angesprochen. Die Felder würden von der internen Optimierungsfunktion ignoriert und wären deshalb dem QSF für die Ausgabe nicht bekannt. (Mit QSF können nur Felder ausgegeben werden, die im Programm definiert sind.)

Data Dictionary Listbild im Programm 2680

Das Listing der Data-Dictionary-Einträge im Programm wird über die folgenden Options-Parameter gesteuert:

CPG Die Eintragung CPG für die Anzeige in RPG-ähnlicher starrer Notation impliziert das Auflisten aller angezogenen Felder.

RPG Siehe CPG.

DIC Es werden alle codierten Statements aufgelistet. Die aus dem Data Dictionary angezogenen Felder werden unter ihrer Satzbestimmung mit einem '-' in Spalte 6 (allerdings im starren Format) angelistet.

ENT Entire Input. Der gesamte Input wird angezeigt. Der Default Suppress wird damit aufgehoben, so dass Eingabestrukturen vollständig angelistet werden, sowohl die vom Programm benutzten als auch die unbenutzten Felder. ENT wird kombiniert mit DIC oder MIX.

FRE Nur das Quellenprogramm im Freien Format wird angelistet. FREE ist default und braucht nicht codiert zu werden.

GEN arbeitet wie CPG. Im Unterschied zu CPG werden aber in der Input Division nicht benutzte Felder auch nicht angezeigt.

MIX Zusätzlich zum Quellenformat werden alle generierten Statements im RPG-Format angelistet, also insbesondere auch die aus dem Data Dictionary angezogenen Feldbeschreibungen.

Beispiel: Besonderheit bei der Output Division

```
- OPTIONS DIC.
:
- -O. FILE CPGWRK DD TYPE F2
-*                        KEY       14
-                         SATZ     100
-     FILE CPGWRK DD TYPE F2 ADD
-                         KEY       14
-                         SATZ     100
```

Wird ein Options-Parameter gewählt, bei dem die aus dem Data Dictionary angezogenen Felder angezeigt werden, so wird bei einem Update in der Output Division das Schlüsselfeld der Datei als Kommentarstatement angelistet, da es nicht zurückgeschrieben werden darf; beim Hinzufügen wird das Schlüsselfeld bzw. werden die (Teil-) Schlüsselfelder normal als Feldbestimmung generiert.

Standardmäßig würden die aus dem Data Dictionary angezogenen Felder in alphabetischer Reihenfolge angezeigt. Durch einen Eintrag in der Standard Header-Karte kann der Systemprogrammierer diese Reihenfolge durch die nach Linienbelegung aufsteigende Sortierung ersetzen. Diese Sortierung ist zu empfehlen.

Testhilfen 2800

QDF - Quick Debugging Facility 2810

Im Lieferumfang des CPG2 ist die interaktive Testhilfe QDF enthalten. Sie ist im Handbuch 'CPG2..Serviceprogramme' ausführlich beschrieben.

QDF löst die Testhilfe-Operationen DEBUG und SDUMP ab. Die Funktionen dieser Operationen sind im QDF implementiert.

Special Terminal Dump 2830

Die Operation 'SDUMP' beinhaltet die Operation 'DEBUG' und kann von beliebiger Stelle des zu testenden Programms aufgerufen werden.

In der zweiten Informationszeile wird unterschiedlich zur Operation DEBUG die Programmadresse, der SDUMP-Code und die zuletzt betätigte Funktionstaste angezeigt.

Als erstes erscheint folgende Maske:

```
**********************************************************************
                          T E S T - H I L F E
**********************************************************************
BEZUGSZAHLEN EIN
70


**********************************************************************
PROG.-ADRESSE 00029A            CODE ABCD          FUNKTIONSTASTE DE
**********************************************************************
```

Nach Betätigung der Datenfreigabetaste erscheint ein Terminaldump mit folgendem Format:

```
Programm = TST005   TCA                                  05.03.93  15.28UHR
E         F         0         1          3         4         5
002DBAA0  0023F180  0000001C  00      0A 502DA80A  502DA80A  502DB80A
502DC80A  502DD80A  802DBBE6  00      00 002D8B08  0023F080  001F3E40
6         7         8         9          B         C         D

00000000  0023F000  00000000         000  0023F080  ..0.....  ........
00000010  00000000  002082C0         000  0023F090  ........  ....d...
00000020  701F6A46  0023F18         BAA0  0023F0A0  ......1.  ........
00000030  502D9B1E  502D980          6952  0023F0B0  ........  ........
00000040  502DC80A  502DD8         2F720  0023F0C0  ..H...0.  ...U..7.
00000050  011EBCA8  001EA2         23300  0023F0D0  ........  ........
00000060  402D9D72  0023F         2DBAA0  0023F0E0  ......1.  ........
00000070  402D9B1E  502D          02DB80A  0023F0F0  ........  ........
00000080  00220095  00           00000000  0023F100  ........  DV......
00000090  00000000  0            0023F739  0023F110  ........  ......7.
000000A0  502D9D00  0            002DBAA0  0023F210  ......1.  ........
000000B0  502D9B1E            0  502DB80A  0023F130  ........  ........
000000C0  502DC80A           E4  4022F720  0023F140  ..H...Q.  ...U..7.
000000D0  0022EF40          000  00000000  0023F150  ..?.....  ........
000000E0  0000000           000  00000000  0023F160  ........  ........
000000F0  000000           0000  00000000  0023F170  ........  ........

SDUMP = ...  +           DRUCKER = ....
```

In der 1. Zeile wird der Phasenname des Programms (hier TST005) und der Bereich angezeigt, auf den der Terminal-Dump erstmals zugreift. Beim 1. Aufruf ist dies immer die Task-Control-Area (TCA).

In den vier folgenden Zeilen wird der Inhalt der Register E,F,0,1 usw bis B,C,D angezeigt, wobei jeweils links über bzw. unter dem Registerinhalt die Registernummer angezeigt wird. Im obigen Beispiel bedeutet:

```
F          Der Inhalt des Registers  15 (F)  war vor  Eintritt  in den
0023F180   SDUMP  = 0023F180.
```

Die folgenden 16 Zeilen geben jeweils 256 Bytes des Hauptspeichers in hexadezimaler und Character-Schreibweise wieder. Die Spalten 2,3,4 und 5 dieses Abschnitts zeigen dabei den Speicherinhalt hexadezimal an.

Die Spalten 7 und 8 zeigen den Inhalt in Klartext, wobei alle nicht druckbaren Zeichen einschließlich der Sonderzeichen mit einem hexadezimalen Wert kleiner als 'C1' durch einen Punkt ersetzt werden.

Spalte 1 dieses Abschnitts zeigt die relative Adresse zum jeweiligen Anfangspunkt, im obigen Beispiel zur TCA-Adresse.
Spalte 6 dieses Abschnitts zeigt jeweils zur relativen Adresse in Spalte 1 die absolute Hauptspeicheradresse an.

Über die Programmfunktionstasten PF8 oder Datenfreigabe können die jeweils nächsten 256 Bytes angefordert werden. Dabei kann im Dump beliebig weit geblättert werden.

Mit der Programmfunktionstaste PF7 kann entsprechend zurückgeblättert werden.

Die letzte Zeile bietet schließlich die Möglichkeit, den Dump von jeder beliebigen Stelle der CPU anzuzeigen. Dabei hat der Programmierer die Möglichkeit, auf folgende Bereiche direkt aufzusetzen:

```
ADR   Adresse XXXXXX
CSA   Common System Area
CIO   CPG Input-Output-Area
CWA   Bereich Common Work-Area
END   Ende SDUMP                  Das Aufsetzen erfolgt durch Einga-
HLB   HL1 Library                 be einer der  nebenstehenden  Kon-
IFC   Interface Com. Area         stanten  bei: SDUMP = ...  und Be-
MBK   Methodenbank                tätigung der Datenfreigabe-Taste.
PRG   Anwendungsprogramm
PWA   Private Work Area
TCA   Task Control Area
TCT   TCT  User    Area
```

Dabei kann der Programmierer bei + ... noch einen Verschiebungsfaktor angeben, z.B. TCA + 000100 ist die Transaction Work Area.

Wird bei SDUMP = ... die Konstante 'ADR' eingetragen, so muss das folgende Feld (+ .....) eine gültige Adresse enthalten. Diese Adresse, die auch außerhalb der TP-Partition liegen kann, gilt dann nach Druck der DE -Taste als Startadresse für den Terminal-Dump.

Bei Drucker = ... wird die Destination-Id des Online-Druckers für den Ausdruck mit PF4 angegeben.

```
Programmfunktionstasten:  DE     Vorwärtsblättern um 256 Bytes.
                          PF2    Neuaufsetzen.
                          PF4    Drucken.
                          PF7    Rückwärtsblättern um 256 Bytes.
                          PF8    Vorwärtsblättern um 256 Bytes.
                          andere Ende SDUMP
```

Einschränkungen bei OPTIONS-Parameter BIG 2920

Die Operationen WRITE, DELET und UPDAT sind nicht unterstützt.

Einschränkungen bei Bausteinen ohne Dataset-Logik 2930

Enthält die OPTIONS-Bestimmung eines HL1-Bausteins nicht die Parameter DAT oder PWA (für Dataset-Logik), dann ist das sequentielle Lesen ohne Schlüssel nicht unterstützt. Die Operationen READ, READ-BACK READ-PAGE und READB-PAGE können dann nur mit Angabe eines Schlüsselfeldes codiert werden.

Siehe auch Tabelle 'Maximalwerte in CPG-Programmen', Seite 7040.

ESA-Mode 2970

Alle softwareseitigen Verbesserungen, die ESA bietet, sind im Command Level verwirklicht. In Zukunft wird der Macro Level nicht mehr unterstützt, genauer

- in VSE/ESA-Umgebungen in den CICS/ESA Releases größer 2.2
- in Z/OS/ESA-Umgebungen ab CICS/ESA Release 3.2

Wenn in diesem Kapitel vom ESA-Mode die Rede ist, dann beziehen sich die Ausführungen immer nur auf die genannten Umgebungen, in denen der Macro Level nicht mehr existiert.

Zur Umstellung auf den ESA-Mode müssen CPG-Programme mit einem neuen Command Level/ESA-Interface umgewandelt werden. Mit dieser Umstellung, (die erst bei Einsatz der oben genannten Umgebungen notwendig wird), kann schon vorab begonnen werden.

ESA-Mode-fähige Programme sind unter älteren CICS-Releases bereits ablauffähig.

Statistik 2972

CPG2-Benutzern empfehlen wir, die CPG2-Statistik zu aktivieren. Somit kann jederzeit überprüft werden, welche Programme bereits ESA-fähig sind.

Options und Standard-H-Karte 2974

Die ESA-Fähigkeit wird über die Options gesteuert. Da es sich um einen unternehmensweiten Standard handelt, wenn ESA-Mode eingeführt werden soll, sollte auch die entsprechende Eintragung (E in Spalte 47 des Copy Books CPGUCSTH) in der Standard-H-Karte vorgenommen werden.

Im Programm heisst der entsprechende Parameter ESA.

ESA erzeugt ein Command Level-Programm, das auf das Command Level/ESA-Interface zugreift.

Macro Level-Programme sowie Nicht-ESA-Command Level Programme werden in den genannten Umgebungen nicht mehr ablauffähig sein.

Da aber bis zur tatsächlichen Umstellung noch eine lange Übergangsphase den Parallelbetrieb von ESA-Mode-fähigen Programmen und Nicht-ESA-Mode-fähigen Programmen ermöglicht, ist dies auch mit CPG möglich.

Der Options-Parameter NON-ESA generiert auch bei einem Parameter E(SA) in der Standard-Header-Karte ein Command-Level-Programm, das auf das (alte) Command Level Interface (CPGCLI) zugreift.

Der Options-Parameter MACro generiert ein Macro Level Programm.

Beachte:

Grundsätzlich überschreiben Options-Parameter, die im Programm gesetzt werden, den Eintrag in der Standard-Header-Karte. Hiervon ausgenommen ist der Parameter COM für Command Level-Programme. COM überschreibt ESA nicht. Um 'alte' Command Level-Programme zu generieren, steht deshalb der Parameter NON-ESA zur Verfügung.

Wird in Spalte 97 der Standard Header-Karte ein 'C' wie CANCEL eingetragen, dann brechen Umwandlungen ab, die nicht zu ESA-Mode-fähigen Programmen führen. Ansonsten bekommen solche Programme eine Warning CPG0070 (siehe unten, Fehlermeldungen).

Linken 2976

Bei der Umwandlung ESA-Mode-fähiger Programme muss beachtet werden, dass das Linken mit INCLUDE DFHEAI und CPGSCEIC durchgeführt wird.

Fehlermeldungen 2977

Im ESA-Mode brechen Macro Level-Programme und nicht-ESA-Mode-fähige Command-Level-Programme mit ASRD ab.

Umwandlungen, die nicht ESA-Mode-fähige Programme generiert haben, werden mit einer WARNING CPG0070 abgeschlossen.

Veränderungen infolge des ESA-Mode 2978

- Einzelsatzverarbeitung von Temporary Storage-Bereichen wird nicht mehr unterstützt sein.

  Abhilfe:

  Durch die Eintragung S in der FILE Description (bzw. im Data Dictionary) erzeugt CPG eine Queue,die aus einem Satz besteht. Sonstige Eingriffe in die Verarbeitung der Queue brauchen vom Programmierer nicht vorgenommen zu werden.

  Hierbei ist zu beachten, dass alle Programme, die den entsprechenden TS-Bereich verarbeiten, gleichzeitig konvertiert werden müssen. Die gemischte Verarbeitung eines Bereichs (einmal als Queue, einmal als Einzelsatz) führt zu einem "ILLOGIC"-Fehler.

- Um aus ESA-Mode-fähigen Programmen HL1-Module aufrufen zu können,

müssen diese Module mit Release 1.6 des CPG (oder höher) umgewandelt sein.

- Um Programme in der CICS-Version zu generieren (das sind Programme, die ohne Methodenbank ablaufen) ist statt des OPTIONS-Parameters CIC der Parameter CICSESA erforderlich.

- Register 13 adressiert nicht mehr die CSA. In Programmen mit assemblergeschriebenen Sequenzen können zwischen BEGAS und ENDAS die Felder CPGCSA und CPGCSACI nicht mehr angesprochen werden.

Abschnitt 3 Programmentwurf 3000

Syntax 3010

Der Programmtext wird in ein 80-stelliges 'Codierformular' eingetragen. Dieses kann auf zwei Arten gegliedert sein:

1. Codierung beginnt ab Stelle 1

   Dazu muss in der ersten Zeile des Programms ab Spalte 1 OPTIONS als Schlüsselwort stehen. Der gesamte Programmcode wird dann im folgenden von Stelle 1 bis Stelle 71 gelesen.

   Alle weiteren Regeln sind unter 2. beschrieben; es muss hier aber beachtet werden, dass zur Zeit nur 71 Stellen für den Code zur Verfügung stehen. Das bedeutet, dass alle weiter unten angegebenen Maximalwerte noch um 8 verkleinert werden müssen; also:

   Letzte gelesene Stelle: Spalte 71 !
   Letzte Stelle, ab der ein Statement beginnen darf: Spalte 64 !

2. Programmcode beginnt ab Spalte 8

   Stelle 6 bleibt frei oder enthält ein Minuszeichen.

   Stelle 7 bleibt grundsätzlich frei.

Der Programmtext besteht aus einzelnen Worten, die durch ein oder mehrere Blanks voneinander getrennt zu Sätzen oder Statements zusammengefügt werden.

Eine Zeile kann ein oder mehrere CPG2-Statements enthalten. Das Ende eines Statements wird durch einen Punkt mit nachfolgendem Blank angezeigt. Wenn in der CPG-Initialisierungs-Tabelle Dezimalpunkt statt Dezimalkomma angegeben wurde, so wird das Statement-Ende durch ein Semicolon (;) angezeigt. Gleiches gilt, wenn als OPTIONS-Parameter das Schlüsselwort 'SEM' oder SEMICOLON angegeben wird.

Ein Satzende-Zeichen ist nur dann erforderlich, wenn mehrere Statements in einer Zeile beschrieben werden. Auch für das letzte Statement in einer Zeile mit mehreren Statements kann das Satzende-Zeichen entfallen. Eine Ausnahme bilden die OPTIONS, bei denen ein Satzende-Zeichen oder das Schlüsselwort 'END' zwingend vorgeschrieben ist.

Die Worte innerhalb eines Statements werden durch mindestens ein Blank voneinander getrennt. Der Abstand zwischen zwei Worten kann beliebig lang sein, solange das Statement in einer Zeile untergebracht werden kann. Das Ende eines Statements wird durch ein Satzende-Zeichen angezeigt, das einem Wort unmittelbar (ohne Blank) folgen muss und dem ein Blank folgen muss. Beginnt das folgende Statement mit einem Stern, so wird der Rest der Zeile als Kommentar gewertet.

Ein Satz oder Statement darf nicht über das Zeilenende hinausgehen, das heisst, das Satzende-Zeichen muss in derselben Zeile stehen wie das erste Wort des Satzes.

Das letzte Statement einer Zeile darf nicht hinter der 72-sten Stelle beginnen.

Gemischte Verwendung starrer und freier Schreibweise 3020

Es besteht die Möglichkeit, freie und spaltengebundene Schreibweise in CPG2-Programmen gemischt zu verwenden.

Es ist nur zu beachten, dass eine Division entweder mit einem Division Indicator im Freien Format oder mit einer Karte im festen Format begonnen werden muss.

Gliederung 3030

Abhängig von der Art der auszuführenden Anweisungen wird das Programm in verschiedene Bereiche (Divisions) gegliedert. Für jede Division gilt eine eigene Grammatik. Ein Programm kann folgende Divisions enthalten:

OPTIONS	Dieser Bereich enthält Anweisungen an den Compiler für spezielle Aufgaben oder Umgebungen.
FILES (Kurzform: -F)	Dieser Bereich beschreibt die im Programm verwendeten Dateien.
DATA DIVISION oder WORKING STORAGE SECTION (Kurzform: -D)	In diesem Bereich werden die im Programm verwendeten Datenfelder (Variablen) definiert.
FORMS	Dieser Bereich beschreibt die Formularsteuerung eines Druckers.
INPUT DIVISION (Kurzform: -I)	In diesem Bereich wird beschrieben, woher die für das Programm erforderlichen Daten geholt werden.
PROCEDURE DIVISION (Kurzform: -C)	In diesem Bereich wird beschrieben,wie die Daten verarbeitet werden sollen. Am Ende der Procedure Division liegen die Subroutines, die optional mit dem Schlüsselwort SUBROUTINES (Kurzform: -SR) vom Hauptprogramm abgetrennt werden können. Zwischen Hauptprogramm und Subroutines können noch Total-Rechenbestimmungen liegen (siehe Kapitel 2270). Diese werden mit Division Indicators -L0 bis -L9 kenntlich gemacht.
OUTPUT DIVISION (Kurzform: -O)	In diesem Bereich wird beschrieben, wohin die im Programm ermittelten Daten ausgegeben werden sollen.

Die angegebene Reihenfolge ist zwingend vorgeschrieben.

OPTIONS 3300

Das Schlüsselwort OPTIONS gibt an, dass die folgenden Schlüsselworte für die Steuerung des Compilers zu interpretieren sind. Die folgenden Schlüsselworte können in beliebiger Reihenfolge aneinander gereiht werden, wobei die einzelnen Worte jeweils durch ein oder mehrere Blanks voneinander getrennt werden. Das Ende der OPTIONS wird durch ein Satzende-Zeichen angezeigt. Es können mehrere Zeilen für die Options benutzt werden.

Beachte:

Wird der Phasenname nicht über Job Control-Statements vorgegeben und soll die Options-Parameterleiste mehrere Zeilen umfassen, dann m u s s 'PHAse xxxxxxxx' in der ersten Zeile der Options codiert werden.

Folgende Schlüsselworte sind möglich:

ADD x oder ADR x. Spezielle Adressierungs-Routine.

Die Zuteilung der Register erfolgt im allgemeinen dynamisch. Der Programmierer kann jedoch für spezielle Anwendungen eine eigene Adressierungsroutine ins Programm einfügen, die vorher unter dem Namen 'CPG*CADX' in die Source Library katalogisiert wird. Das Zeichen '*' steht hierbei für den Release-Suffix (z.B. D für 1.4).

Das 'x' in der letzten Stelle wird durch den Eintrag bei ADD x in der OPTIONS-Karte bestimmt. Wird beispielsweise ADD 1 eingetragen, so wird das Copy-Book 'CPG*CAD1' ins Programm eingefügt.

Damit kann der Programmierer beispielsweise die TWA von 4 auf 8 oder 12 K erweitern, oder bestimmte z.B. in eigenen Unterprogrammen benutzte Register von der Adressierung ausnehmen.

Für folgende Eintragungen werden standardmäßig Copy-Books mitgeliefert, d.h. diese Werte kann 'x' annnehmen:

'B' CPG intern.
'C' CPG intern.
'D' DL/I Anwendungsprogramme.
'E' ETC Anwendungsprogramme.
'G' CPG intern.
'I' CPG intern.
'J' CPG intern.
'K' CPG intern.
'L' CPG intern.
'O' OS Assembler.
'P' CPG intern.
'R' CPG intern.
'S' Shadow Anwender.
'T' TCSS Anwender.
'U' PLT-Programm mit PWA-Verwendung.
'V' Com.-Level und DL/I-Dataset.
'W' siehe 'V' und TWA Size 8K.

'X' CPG intern.
'Y' CPG intern.
'#' Command Level Anwendungsprogramme (#=X'7B').
'0' Com. Level max. Programmgröße 20K und TWA SIZE 8K
'1' Macro Level max. Programmgröße 20K und TWA Size 8K
'2' CPG intern.
'3' 12K TWA in Batch-Programmen
'4' CPG intern.
'8' PLT-Programm mit PWA-Verwendung (Command Level).
'9' DL/I Anwendungspr.mit max. Prog.20K und TWA Size 8K.

Beispiel: 'ADD D' = 'ADDRESS D' = 'ADDRESS DL/1'

ASM x Soll der CPG-Source-Code nicht mit //EXEC ASSEMBLY in Maschinensprache übersetzt werden, sondern mit einer anderen Assemblierprozedur, so wird hier statt x der Suffix des Prozedurnamens angegeben.

Es muss dann sichergestellt sein, dass es eine Assemblierprozedur mit dem Namen CPGUASSx gibt, die dann wie folgt generiert wird: // EXEC PROC=CPGUASSx.

Beispiel: OPTIONS ASM-SUFFIX H für die Prozedur CPGUASSH

ASS x oder ASSEMBLER-LIST x ist die Art der Assembler-Liste, wobei 'x' folgende Werte annehmen kann:

'A' Assemblerliste ohne Makro-Auflösung.

Der für die Verarbeitung wesentliche Teil des Assemblerprogramms wird gedruckt. TP-Dummy-Sections, Makros und CPG-Unterprogramme werden nicht gedruckt.

'C' Komplettes Assemblerprogramm.

Das vom CPG erzeugte Assemblerprogramm wird komplett angelistet.

'D' Assemblerliste mit allen Dummy-Sections.

Wie 'A', jedoch werden alle Dummy-Sections mit angelistet.

'M' Assemblerliste mit Makroauflösung.

Wie 'A', jedoch werden zu jedem Makro die generierten Statements mit angelistet.

'N' Keine Assemblerliste

Das vom CPG erzeugte Assembler-Programm wird nicht aufgelistet. Es wird nur die CPG-Umwandlungsliste einschließlich CPG- und Assembler-Diagnostik gedruckt.

'S' Komplettes Assemblerprogramm mit Short-Cross-Reference.

Wie 'X', jedoch mit Anlistung der Assembler Short-

Cross-Reference (nur VSE).

'T' Transaction Work Area.

Von der Assemblerliste wird nur die Transaction-Work-Area und die Assembler-Diagnostik angelistet.

'X' Komplettes Assemblerprogramm mit Cross-Reference.

Wie 'C', jedoch zusätzlich mit Anlistung der Assembler Cross-Reference.

X kann auch der erste Buchstabe eines Wortes sein.

Beispiel: ASSEMBLER-LISTE TWA

ATT C oder ATTRIBUTE C.

Ein 'C' bewirkt, dass die Eintragungen nach dem Schlüsselwort ATT in der Output Division als Hardware-Attribute interpretiert werden. Damit steht dem Programmierer der gesamte Hardware-Attribut-Satz des Bildschirms zur Verfügung, wenn aus irgend einem Grund die CPG-Attribute nicht ausreichen.

C kann auch der erste Buchstabe eines Wortes sein.

Beispiel: ATT CICS

AUT Bei Programmverbindungen soll ein automatisches RNDOM*ALL gegeben werden.

BAT Das generierte Programm soll in einer Batch-Partition ablaufen.

BIG Programm ist größer als 24 K.

Dieser Parameter bewirkt, dass für jede Eingabe-Datei,für jede Subroutine, für die Procedure und Output Divisions und für die Feldaufbereitung jeweils eine eigene CSECT angelegt wird.

Für diese CSECTS gelten folgende Einschränkungen:

- Pro Programm sind maximal 200 CSECTS unterstützt.
- Die CSECT für die Feldaufbereitung darf maximal 4K umfassen; höchstens 400 Felder können pro Feldaufbereitung übertragen werden.
- Der allgemeine Programmteil (Procedure Division ohne Subroutines) darf nicht größer werden als 8 K.
- Alle anderen CSECTS können maximal 12 K groß sein.
- Die TWA Size ist auf 4 K begrenzt, kann aber über den Parameter ADD auf 8 K erhöht werden (Siehe oben). Siehe 12K.

CAT oder CATAL. OPTION CATAL- und PHASE-Karte werden über Job Control-Statements vorgegeben.

CIC oder CICS/CICSE. Das Programm arbeitet dann ohne die Methodenbank des CPG (CICS-Version). Schlüsselwort CICS generiert ein

Macro Level-Programm, CICSEsa ein Command Level-Programm.

COL oder COLUMN. Es kann für eine beliebige Spalte der generierten H-Karte ein beliebiger Character angegeben werden in der Form: COL 47 - oder COLUMN 47 = '-'.

Das Minuszeichen im Beispiel wird gesetzt, um einen Eintrag der Standard-Options zu überschreiben.

COM oder COMMAND LEVEL. Das generierte Programm soll unter CICS Command-Level ausgeführt werden.

CPG listet nur die generierten CPG-Statements, nicht aber die vom Programmierer codierten '-' Karten an.

DAT für Dataset Logik. Für HL1-Module kann dieser Eintrag vorgenommen werden, um die PWA des Moduls innerhalb einer Task zu erhalten. Diese Verarbeitungsart bietet neben Performance-Vorteilen den Service, dass die Files Division eines solchen Moduls nicht vollständig im zugehörigen Hauptprogramm enthalten sein muss.

DDL gibt die langen Feldnamen aus dem Data Dictionary in der Umwandlungsliste aus.

DDS gibt die kurzen Feldnamen aus dem Data Dictionary in der Umwandlingsliste aus.

DEB für DEBUG. Soll das Debug Facility QDF genutzt werden, so ist Voraussetzung, dass das Programm mit dem Options-Parameter DEBug umgewandelt wird.

Zu beachten ist, dass sich durch den Parameter DEBug der Programmcode um 8 Bytes pro Statement vergrößert.

DEC für DECK. Es wird ein Object Deck gestanzt.

DEF für define. Alle in Data Division und Input Division beschriebenen Felder werden für das Programm definiert, unabhängig davon, ob sie im weiteren Programmablauf noch benötigt werden. Die Optimierungsfunktion des CPG wird durch DEF also aufgehoben.

DIC Es werden die programmierten Statements einschließlich der Data Dictionary-Eintragungen als Statements in freier Schreibweise aufgelistet (Spalte 6 : '-').

END Ende der OPTIONS.

ENT für entire Input. Es wird eine vollständiege Liste der Input Division angezeigt, nicht nur die von CPG im Programm tatsächlich genutzten Felder. ENT wird kombiniert mit den Optionen DIC oder MIX.

ESA ESA-Mode-fähige Programme. Siehe Kapitel 2970.

GEN listet die vom Compiler generierten Befehle an und unterdrückt das Listing nicht benutzter Felder in der Input Division.

HL1 X Für X wird der Index der privaten HL1-Library eingetragen.(=LIB)

INT für Interrupt. Der Parameter bewirkt, dass bei einem Programm-

fehler (Cancel) in einem Batch-Programm eine Programmunterbrechung erfolgt.

LAN X oder LANGUAGE XXXXX. X kann folgende Werte annehmen:

Ein 'D', 'E', 'I', 'J' bewirkt, dass die Ausgabe sämtlicher Texte in englischer Sprache erfolgt.

Ein ' ' oder ein 'G' bewirkt, dass die Ausgabe sämtlicher Texte in deutscher Sprache erfolgt.

Jedes andere Zeichen bewirkt, dass eine vom Programmierer selbst unter CPGSX* katalogisierte Textphase angezogen wird, wobei der * durch das eingetragene Zeichen ersetzt wird.
Die Eintragung hat keine Auswirkungen mehr auf die Darstellung der Dezimalzeichen.

Beispiel: LANGUAGE FRENCH

LIB X Für X wird der Index der privaten HL1-Library eingetragen.(=HL1)

LIS X Listaufbereitung. X kann folgende Werte annehmen:

E	Fehlermeldungen werden rechts neben dem Text gedruckt
F	Ein Blockschaltbild wird gedruckt
L	Die Zeilennummer am linken Rand wird unterdrückt
N	Die Programmiererchecklist wird unterdrückt
O	Die Fehlermeldungen werden zwischen den Zeilen gedruckt
P	Zu jedem generierten Statement wird die Zahl der erforderlichen Bytes gedruckt.

LON Long Array Names. LONG (oder Standard-Header Spalte 100) bietet die Möglichkeit, mit beliebig langen Feldgruppennamen zu arbeiten. Zu beachten ist dabei, dass der Name des Indexfelds nur eine Stelle lang sein darf. Es werden auch 5- und 6-stellige Feldgruppennamen in CPxx-Namen intern umgesetzt.

LOW Kleinbuchstaben werden nicht automatisch übersetzt.

MAC Ein Macro-Level-Programm wird generiert (Überschreiben eines Eintrags im Standard Header, siehe auch Kapitel 2970).

MAI oder MAIN (nur für HL1) Es soll ein HL1-Hauptprogramm generiert werden.

MAP Bei Verwendung von MAP-Befehlen sollen die Maps in der Reference Liste aufgelistet werden.

MIX Es werden sowohl die programmierten als auch die generierten Statements aufgelistet.

MVS Das Programm soll unter OS oder z/OS laufen.

NON- NON-ESA Command Level-Programme (zum Überschreiben des Standard Headers, so dass Programme mit dem CPGCLI ausgeführt werden). Siehe auch Kapitel 2970.

NOS NO SYSIN, ermöglicht die CPG-Umwandlung über die Punch Queue, so dass IJSYS04 nicht benötigt wird. Für diese Art der Umwandlung wird der Job CPGZPUN benötigt, der unter Zubehör in Kapitel 7000 beschrieben ist.

OPT Optimierung für numerische Operationen. Dieser Eintrag bezieht sich auf die numerischen Operationen '+', '-' und '='. Optimiert bedeutet, dass zu diesen Operationen direkter Assembler-Quellencode generiert wird. Somit wird bei der Ausführung nicht in die CPG-Methodenbank verzweigt. Insbesondere in CPG-Batchprogrammen bewirkt dies eine erhebliche Verbesserung der Laufzeit.

Beachte:

Die Parameter OPT und DEBug schließen sich gegenseitig aus. Nur einer der beiden Parameter ist sinnvoll. CPG2 entscheidet sich für den zuletzt genannten.

PHA XXXXXX oder PHASE XXXXXX. XXXXXX = Phasen-Name. Phasen-Namen dürfen maximal acht Stellen lang sein.

PUN oder PUNCH. Es wird ein Assembler Deck gestanzt.

PWA für PWA erhalten. Für HL1-Module kann dieser Eintrag vorgenommen werden, damit die PWA innerhalb einer Task bestehen bleibt, aber bei jedem Aufruf erneut initialisiert wird (ohne Initialisieren: DAT). Diese Verarbeitungsart bietet neben Performancevorteilen den Service, dass die Files Division eines solchen Moduls nicht vollständig im zugehörigen Hauptprogramm enthalten sein muss.

QSF Bei Verwendung von MAP-Befehlen sollen die Maps und die von der Map benutzten Felder in der Reference Liste angelistet werden.

QLF Bei Verwendung von List-Befehlen sollen die List-Dokumente und die darin benutzten Felder in der Umwandlungsliste ausgegeben werden.

QTF Siehe QLF.

RDR oder REAder zur Angabe der Leser Adresse (in der Regel ein Eintrag in der Standard H-Karte CPGSTH.)

ROO oder ROOT (nur für HL1) Es soll ein HL1-Hauptprogramm generiert werden.

RPG listet nur die generierten CPG-Statements, nicht aber die vom Programmierer codierten '-' Karten an.

RUN Assembler Umwandlung wird auch bei CPG-Fehlern ausgeführt

SEM oder SEMICOLON. Das Ende eines Statements wird durch Semicolon angezeigt.

Beachte:

SEM darf nicht das letzte Schlüsselwort der Options sein.

SHA Shared Data. Der Eintrag wird für HL1-Module vorgenommen, die automatisch (d.h. ohne Angabe eines Datenkanals) mit dem rufen-

den Programm Daten austauschen sollen.

Der Datenaustausch findet statt zwischen Feldern, die in beiden Programmen gleiche Namen und gleiche Feldeigenschaften besitzen. Die Datenübergabe erfolgt wie bei QPG, nur bis zum Feld CPGEDS, wenn dies im Modul definiert ist.

SHO Als Gegenstück zu LONG (s.o.) sorgt SHOrt dafür, dass der Compiler 5- und 6-stellige Feldgruppennamen intern nicht übersetzt

SIG oder SIGN. Zone C für numerische Felder (vgl. Kapitel 2145)

SIZ XXX oder SIZE XXX. XXX ist die Size Eintragung für die Programm-Umwandlung und muss 3 Stellen groß sein.
Size 096 K.

SUB für Subprogramm. Dieser Parameter bewirkt bei Batchprogrammen, dass die Ausführung bei Programmende an ein übergeordnetes Programm (z.B. DL1) zurückgegeben wird.

TIT XXXXXXX oder TITLE XXXXXXX setzt den nachfolgenden Titel (XX..XX), der maximal 23 Stellen lang sein darf, als Überschrift über alle Seiten der Umwandlung. Blanks innerhalb des Titels sind durch '#' zu ersetzen. (#) = Nummernzeichen (Doppelgitter).

Beispiel: TITLE DIES#IST#EIN#TITEL

TRA für Trace. Soll Quick Debugging Facility QDF genutzt werden, so kann statt des Parameters DEBug der Parameter TRAce angegeben werden. TRA bewirkt, dass nur die numerischen Operationen im interaktiven Test verfolgt werden können.

Vorteil von TRA ist, dass sich der Programmcode im Gegensatz zu DEBug nicht vergrößert.

TWA XXXX. XXXX = TWA Größe.

Wird von einem anderen Programm in dieses Programm verzweigt, so wird hier vierstellig die TWA-Größe des aufrufenden Programms eingetragen. Kann das Programm von mehreren Programmen aufgerufen werden, so ist die TWA-Größe des größten aufrufenden Programms einzutragen.
Siehe auch Programmverbindungen.

Wenn nur ein Teil der TWA übernommen werden soll, so ist der dezimale Wert aus der TWA-Liste einzutragen,der hinter dem entsprechenden Feld aufgeführt wird. (Minimum = 116 Bytes, Maximum = 7836 Bytes).

Numerische Felder,die hinter dem übernommenen TWA - Bereich beginnen, werden auf X'0C' initialisiert.

USE Bei Programmverbindungen soll k e i n automatisches RNDOM *ALL gegeben werden.

XRE oder XREF druckt eine CPG-Cross-Reference-Liste.

12K Dieser Eintrag bewirkt, dass ein Programm generiert werden kann, das größer als 24 K ist und dessen TWA 12 K groß ist.

Es gelten die Einschränkungen des Parameters BIG sinngemäß.

Zu beachten ist, dass 'TWA' hier nicht als Dummywort eingesetzt werden kann, da TWA selbst ein Schlüsselwort ist.

Verbindende Worte im Text sind erlaubt, soweit sie nicht mit Schlüsselworten identisch sind. Wir empfehlen den verbindenden Text aber nicht, da es möglich ist, das in späteren Releases diese Füllwörter eine Bedeutung bekommen.

```
OPTIONS TITLE BEISPIEL THIS IS A COMMAMD LEVEL PROGRAM.
```

ist identisch mit: `OPTIONS TIT BEISPIEL COM.`

Weitere Beispiele für OPTIONS:

```
- OPTIONS BIG RUN LIS E MIX TIT TEST PHA EXAMPLE2.

- OPTIONS PHASE EXAMPLE3
-         COMMAND-LEVEL
-         SEMICOLON
-         TITLE BUCHUNGEN
-         QSF
-         END

- OPTIONS PHASE EXAMPLE3
-         ENT DIC                * vollständiges Data Dictionary
-         12K                    * 12 K TWA, beliebig großes Programm
-         TITLE BUCHUNGEN
-         QSF.                   * Masken und Felder werden angelistet
```

Beachte:

Der Phasenname muss in der ersten Zeile der Options codiert werden.

Options und Standard-Header CPGSTH 3350

Vom Systemprogrammierer wird eine sogenannte Standard-Header-Karte angelegt, die alle für die Umgebung standardisierten Informationen enthalten sollte. Diese Standards werden gegebenenfalls von der Options-Anweisung überschrieben.

Dazu zwei Besonderheiten:

1. Ausnahme: Die Eintragung für ESA-Umwandlungen (CPGSTH Spalte 47) hat Vorrang vor dem entsprechenden Options-Eintrag !

2. Gibt es kein Options-Schlüsselwort, um einen Eintrag im Standard-Header zu überschreiben, so muss man in diese Spalte ein Minuszeichen setzen, um die Funktion aufzuheben.

   Wie jedes andere Zeichen kann auch das Minus mit dem Parameter COL

gesetzt werden: OPTIONS COLUMN 46 = '-'.

Die folgende Tabelle dient der Zuordnung der Options-Schlüsselworte zu den Spalten des Standard-Headers CPGSTH.

OPTION		Spalte(n)	Eintrag
Default	Liste im CPG2-Format	42	F
Default	Liste ohne Anzeige der optimierten Felder	16	S
Default	HL1-Modul	51	H
ADD x	Adressierung	48	x
ADR x	Adressierung	48	x
ASM x	zur Nutzung von High Level Assemblern	8-9	Ax
ASS x	Umfang der Assembler-Liste	11	x
ATT x	Attribute	49	x
AUT	automatisches RANDOM *ALL	33	Y
BAT	Batchprogramm	47	B
+	Batchprogramm (ist immer Hauptprogramm)	51	C
BIG	Beliebig großes Programm	32	S
CAT	Phase über // OPTION CATAL vorgegeben	31	O
CIC	Programm läuft ohne Methodenbank (Macro Level)	47	C
CICSESA	Programm läuft ohne Methodenbank (ESA)	47	D
COM	Command-Level Programm	47	L
CPG	Liste im CPG1-Format (RPG-Syntax)	16,42	*
DAT	Dataset-Logik im HL1-Modul	34	D
DDL	Lange Namen aus dem Data Dictionary in der Liste	102	L
DDS	Kurze Namen aus dem Data Dictionary in der Liste	102	-
DEB	Debugging mit QDF möglich	46	S
DEC	Object Deck stanzen	10	C
DEF	Definiere alle Input-Felder	16	*
DIC	Dictionary auflisten	42	D
ENT	Dictionary vollständig auflisten (ENTire DIC)	16	*
ESA	ESA Command-Level-Programm	47	E
GEN	Generierte Statements	42	C
HLI x	Private HL1-Library	22	x
HL1 x	Private HL1-Library	22	x
INT	Interrupt in Batch-Programmen	46-47	IB
LAN x	Language-Parameter	21	x
LIB x	Private HL1-Library	22	x
LIS x	Aufbereitung des Listings	15	x
LON	Lange Feldgruppennamen (auch für 5-/ 6-stellige)	100	A
LOW	Groß-/Kleinschreibung in taskorientierten Prog.	39	L
MAC	Macro-Level-Programm	47	M
MAI	Hauptprogramm (Main)	51	C
MAP	Maps und QLF-Listen in der Liste	46	M
MIX	Generierte Statements und codierte gemischt	42	M
Z/OS	Umwandlung für eine Z/OS-Umgebung	8-9	OS
NON-	Non-ESA Mix Mode	47	O
NOS	NO Sysin, Umwandlung über Puncher statt IJSYS04	10	P
OPT	Optimierung numerischer Operationen	46	O
PUN	Stanzen	10	D
PWA	Dataset-Logik im HL1-Modul, PWA initialisiert	34	S
QLF	Maps und QLF-Listen mit Dokumentation im Listing	46	N
QSF	Maps und QLF-Listen mit Dokumentation im Listing	46	N
QTF	Maps und QLF-Listen mit Dokumentation im Listing	46	N
RDR xxx	Reader-Adresse	12-14	xxx
REA xxx	Reader-Adresse	12-14	xxx
ROO	Hauptprogramm (Root Phase)	51	C

RPG	Liste im CPG1-Format (RPG-Syntax)	16,42	*
RUN	Umwandlung auch im Fehlerfall ausführen	50	A
SHA	Shared Data (EXHM ohne Kanal-Angabe)	34	*
SHO	5- und 6-stellige Arraynamen bleiben unverändert	100	-
SIG	Sign	40	S
SIZ xxx	Size	7-9	xxx
SUB	Batch Subprogramm	51	S
TRA	Trace mit QDF nur bei numerischen Operationen	46	T
TWA xxxx	TWA-Länge des Vorprogramms bei EXPR (etc.)	27-31	xxxx
USE	kein automatisches RANDOM *ALL (User macht das)	33	N
XRE	Cross Reference im Anschluß an die Liste	16	*
12K	Beliebig großes Programm mit bis zu 12K TWA	32	T
Default	zur Information:	31	O
Default	zur Information:	49	E

FILES 3400

Vor der ersten Anwendung sollte eine Datei zunächst im Data Dictionary des CPG2 beschrieben werden.

In diesem Fall reicht die Angabe

- FILE PLATTE
- FILE PLATTE DD TYPE01
- FILE PLATTE DD TYPE01 INPUT

um die Datei PLATTE zu definieren.

Werden mehrere Dateien angesprochen, so können sie, soweit die Dateibeschreibung im Data-Dictionary oder in der Standard-File-Tabelle ent-enthalten ist, auch in einem Statement zusammengefasst werden. Als Schlüsselwort wird in diesem Falle 'FILES' verwendet.

- FILES BILD KUNDEN ARTIKEL.

Ist die Datei nicht im Data Dictionary beschrieben,so muss sie an dieser Stelle vom Programmierer manuell beschrieben werden. Dabei müssen die folgenden Parameter in der vorgeschriebenen Reihenfolge angegeben werden. Verbindende Text-Worte sind hierbei nicht erlaubt.

KW DN (IO) (MO) (FO) (BL) (SL) (KL) (FT) (FO) (SV) UN (BA)

KW = Schlüsselwort FILE
DN = Dateiname
IO = Ein-/Ausgabeart
MO = Modus: Queueing für Temporary Storage, Buffer Mode
FO = Satzformat: fest oder variabel
BL = Blocklänge
SL = Satzlänge
KL = Schlüssellänge
FT = Filetyp (KEY, RBA)
FO = Dateiorganisation
SV = Service: z.B. variable Verarbeitung
UN = Einheit
BA = Batch-Erweiterungen

Für die verschiedenen Einheiten gelten sehr unterschiedliche Syntaxregeln. Jede Datei kann aber nach dem gezeigten Schema beschrieben werden, indem man für nicht benötigte Einträge ein '#' (Nummernzeichen) einsetzt.

In der Regel wird man aber die verkürzte Schreibweise wählen, die in Abschnitt 3450 detailliert beschrieben ist.

Schlüsselwort 'FILE'

FILE muss eingetragen werden.

Dateiname

Name der in der TP-File-Table definierten Datei. Der Name darf für VSE 7 Stellen und für z/OS 8 Stellen lang sein.

Das erste Zeichen des Namens muss ein Buchstabe oder ein $-Zeichen sein, für die weiteren Zeichen sind Buchstaben, Ziffern und das $-Zeichen unterstützt.

Wird eine Plattendatei zum ersten Mal benutzt, so muss sie vorher in der TP-File-Table angelegt werden.

Dateiart

Mögliche Eintragungen sind:

'I' Eingabe-Datei (oder INPut)

'O' Ausgabe-Datei (oder OUTput)

'U' Update-Datei (oder UPDate)

'C' Kombinierte Datei (oder COMbined)
Bildschirm-Einheiten im Dialogbetrieb

Diese Eintragungen sind auch dann unterstützt, wenn eine Datei bereits vollständig im Data Dictionary beschrieben ist. Sie überschreiben in diesem Fall den vom Data Dictionary vorgegebenen Eintrag.

Temporary Storage-Verarbeitungsart

'Q' oder 'QUEue' ist eine Besonderheit bei der Temporary Storage-Verarbeitung. Die Eintragung bewirkt, dass im temporären Datenbestand Daten hinzugefügt werden können, also dass der Temporary Storage 'satzweise' verarbeitet werden kann.

Die Verarbeitung von Queues und TS-Einzelsätzen ist in der Syntax unterschiedlich. TS-Einzelsätze sind im manchen Umgebungen gar nicht unterstützt. Soll mit den Befehlen der Einzelsatzlogik eine Queue erstellt und verarbeitet werden, so steht dafür der Eintrag 'S' zur Verfügung.

Druckerart (wahlweise)

Die Eintragung 'B' oder 'BUFfer' bewirkt, dass der Drukker im Buffer-Mode arbeitet. Die Druckausgabe wird dann wie eine Bildschirmausgabe beschrieben.

Satzformat

Mögliche Eintragungen sind:

'F' oder 'FIX' für feste Satzlänge

'V' oder 'VAR' für variable Satzlänge (aber nur bei Bildschirmen und VSAM-Dateien)

Bei fehlender Eintragung wird bei Plattendateien 'F' und Bei Bildschirmdateien 'V' angenommen.

Blocklänge

Platten-Datei:

In diesem Feld wird bei geblockten Sätzen die Blocklänge eingetragen.
Bei einer VSAM-Datei wird die doppelte Satzlänge bzw. Blank eingetragen (in der CICS-FCT Recform=Blocked).

Dataset:

Für Datasets bleibt dieses Feld blank. ('#' muss aber in diesem Fall eingegeben werden.)

Bildschirm:

In diesem Feld wird die Anzahl der Bildschirmzeilen eingetragen. (z.B. 12 , 24 , 27 , 32 , 43, 50 oder 62). Das Feld Satzlänge muss dabei jedoch die Länge einer Bildschirmzeile enthalten.

Satzlänge

Bei der Platten-Datei m u s s die logische Satzlänge eingetragen werden.

Bei Dateien mit variabler Satzlänge muss die Länge eingetragen werden, die dem längsten Satz entspricht, der verarbeitet werden kann.

Bei Datasets m u s s die Länge des Datasets eingetragen werden.

Beim Bildschirm wird die Länge einer Bildschirmzeile eingetragen. Zum Beispiel '40', '80', '132'; für 3290 Bildschirme ist ebenfalls '106' und '160' erlaubt.
Der CPG-Übersetzer überprüft aufgrund dieser Längenangabe die erste und letzte Stelle in den Ein- und Ausgabebestimmungen für die Datei.

Schlüssellänge

Für KSDS-Dateien und Datasets m u s s hier die Länge des (gepackten oder ungepackten) Schlüssels in Bytes eingetragen werden.

Für VSAM ESDS oder RRDS-Dateien sind 4 Bytes anzugeben.

Satzadressierung

Für ISAM- und VSAM-Dateien:

' '	KEY	ISAM und VSAM KSDS und RRDS
'K'	KEY	ISAM und VSAM KSDS und RRDS
'R'	RBA	VSAM ESDS

Bei fehlender Eintragung wird 'K' angenommen.

Dateiorganisation

Mögliche Eintragungen sind:

' '	ISAM-, VSAM- oder DA-Datei
'I'	ISAM-, VSAM- oder DA-Datei
'V'	VSAM-Datei
'L'	VSAM-Datei Locate Mode
'R'	Reuse. VSAM-Datei wird neu geladen (nur Batch)
'AUX'	Auxiliary Storage (auf Platte ausgelagert)
'IND'	Independent Storage (terminal-unabhängig)

Verarbeitungsform

Die Eintragung 'V' oder 'VARiabel' beschreibt einen variablen Dateinamen, allerdings nur für

- Drucker
- Transient Data
- Temporary Storage

Ist hier ein 'VAR' eingetragen, so kann der Dateiname während der Programmdurchführung geändert werden. Der Standardwert ist der hinter dem Schlüsselwort FILE angegebene Name.

Zur Änderung des Dateinamens stehen CPG-interne Felder zur Verfügung, die mit dem alternativen Namen im Programm gefüllt werden:

CPGDID	4-stellig alphanumerisch für Einheit PRINTER
CPGTDI	4-stellig alphanumerisch für Einheit TRANSDT
CPGTSN	8-stellig alphanumerisch für Einheit STORAGE

Ein/Ausgabe-Einheit

Mögliche Eintragungen sind:

DATASET	Datensatz (siehe datenbankunabhängige Programmierung).
DISK	Platteneinheit (für alle Plattentypen) (für alle Dateiorganisationen)
DISPLAY	Bildschirm
DLI	DL/I-Datenbank
ESDS	VSAM-ESDS-Datei
HL1	HL1-Dataset (für CPG3-Anwender)
HL1DS	HL1-Dataset (für CPG3-Anwender)
KSDS	VSAM-KSDS-Datei

PRINTER	Drucker
PUNCHER	Stanzer (nur im Batch)
READER	Leser (nur im Batch)
RRDS	VSAM-RRDS-Datei
STORAGE	Zwischenspeicher (Temporary Storage)
TABLE	Tabelle (vgl. Operation FIND)
TAPE	Band (nur im Batch)
TRANSDT	Übergangsspeicher (Transient Data)
VBOMP	VBOMP-Datenbank

Bildschirm und Drucker können sowohl für Lokal- als auch für Remoteanschluss verwendet werden.

Im Batchbetrieb sind folgende Einheiten nicht zugelassen

DATASET, DISPLAY, TRANSDT, VBOMP

Erweiterungen für die Batch-Programmierung:
(Siehe auch 'verkürzte Schreibweise')

Bei der Batch-Programmierung sind weiterführende Funktionen unterstützt, die hinter der Einheit in beliebiger Reihenfolge angegeben werden können.

Die Bedeutung der Schlüsselbegriffe im einzelnen:

NO OPEN — Dateien mit dieser Eintragung werden nicht automatisch eröffnet, müssen also zur Verarbeitung im Programm explizit mit dem Befehl OPEN eröffnet werden.

Die Eintragung ist nicht unterstützt für die Einheiten DL1, PRINTER, PUNCHER, READER, STORAGE und TABLE.

NO REWind
NORewind — Bänder werden standardmäßig nach der Verarbeitung zurückgespult. Bei No Rewind entfällt das Rückspulen.

STAndardlabel — Diese Eintragung muss vorgenommen werden, wenn sequentielle Banddateien mit Kennsatz verarbeitet werden; ansonsten entfällt die Prüfung der TLBL-Parameter.

SYSnumber — Bei Bändern kann eine sechsstellige SYS-Nummer angegeben werden. Bei Druckern kann ebenfalls eine SYS-Nummer oder SYSLST angegeben werden; damit können in einem Programm mehrere Druckausgaben gleichzeitig erzeugt werden.

UNLoad — Bei der Bandverarbeitung bewirkt UNLoad, dass ein Band zu Programmanfang auf den Bandanfang zurückgespult und am Programmende aus der Bandstation entladen wird.

UNOrdered
UNSortiert — Eine VSAM-KSDS-Datei kann mit dieser Eintragung unsortiert geladen werden bzw. es wird unsortiertes Hinzufuegen ermöglicht.

Es sollte allerdings beachtet werden, dass diese Verarbeitungsart die Performance in hohem Maße beeinträchtigen kann. Größere Datenmengen werden sinnvollerweise sortiert in eine KSDS-Datei eingespielt.

Beispiele für Datei-Beschreibungen:

```
- FILE KUNDEN.
- FILE PSBNAM DL1.
- FILE DR01 OUT FIX 132 PRINTER.
- FILE DR02 OUTPUT BUFFER FIX 120 PRINTER.
- FILE TS01 O F 500 STORAGE.
- FILE TS02 UPD FIX 56 VAR STORAGE
- FILE TS03 UPD  S  FIX 22 VARIABLER AUXILIARY STORAGE
- FILE TS04 UPD QUEUE FIX  INDI STORAGE
- FILE STOR INPUT QUEUE FIX 1000 STORAGE.
- FILE BILD O V 24 80 DISPLAY.
- FILE TAB1 INP FIX  19  9  TABLE.
- FILE TD01 I   F   560  VARIABLER TRANSDT.
- FILE TEST01 OUT FIX 1000 500 05 KEY  REUSE  KSDS UNORDERED UPDATE.
- FILE TAPEIN INP FIX  300  TAPE  STANDARDLABEL SYS010.
- FILE TAPEOUT O  F    300  TAPE  NO REWIND  NO OPEN.
- FILE LISTE2 OUT FIX      132        PRINTER SYS012
- FILE HQTFC  UPD FIX  500   4  HL1
```

Besonderheiten für DISK und die VSAM-Dateiarten:

Bei den Einheiten ESDS, KSDS und RRDS werden zwei numerische Werte erwartet: Der erste Wert ist die Satzlänge, der zweite die Schlüssellänge.

DISK ist für sequentielle Batchdateien vorgesehen. Der erste numerische Wert wird hier als Blocklänge, der zweite als Satzlänge interpretiert. Bei fehlender Blocklänge muss ein Nummernzeichen eingegeben werden.

Werden sämtliche Einträge wie oben beschrieben vorgenommen, so kann auch online mit der Einheit DISK gearbeitet werden:

```
- FILE PLATTE I F 1000 500 05 K I DISK.
- FILE PLATTE INPUT FIX 1000 500 5 KEY INDEX DISK.

- FILE SEQBAT OUT FIX 500 100 DISK.    * sequentielle Batchdatei

- FILE DATEINS  UPD VAR 500 05  KSDS
- FILE DATZWEI  OUT VAR 256 04 RBA  ESDS
```

Verkürzte Schreibweise 3450

In der Regel wird man nur die Parameter angeben, die zur Definition der entsprechenden Einheit notwendig sind.

1. Nur Dateiname und Einheit müssen angegeben werden bei DLI / DL1.

2. Dateiname, Ein-/Ausgabeart, fix oder variabel, die Satzlänge und die Einheit genügen bei den Einheiten:

DISK	Nur für ungeblockte sequentielle Batchdateien (!)
ESDS	
LU61, LU62	
L3286	Zusätzlich ist B für Buffer Mode unterstützt.
PRINTER	
READER	
RRDS	
STORAGE	Die File-Definition für temporäre Speicherbereiche kann bei Bedarf um die Einträge für variable Verarbeitung , Auxiliary- oder Independent-Verarbeitung und Queueing erweitert werden.
TAPE	bei ungeblockten Bändern.
TRANSDT	

3. Dateiname, Ein-/Ausgabeart, fix oder variabel, zwei Längenangaben und die Einheit genügen bei den folgenden Einheiten. In Klammern ist beschrieben, für welche Einträge die beiden Längenangaben angenommen werden:

DISK	Block- und Satzlänge (für sequentielle Batchdateien)
DISPLAY	Block- und Satzlänge
HLI,HL1(DS)	Satz- und Schlüssellänge
KSDS	Satz- und Schlüssellänge
RRDS	Satz- und Schlüssellänge
TABLE	Satz- und Schlüssellänge
TAPE	Block- und Satzlänge (bei geblockten Bändern)

Erweiterungen für den Batch-Betrieb 3455

Erweiterungen für den Batch-Betrieb wie NOREWIND, NO OPEN, etc. können direkt hinter dem Dateinamen angegeben werden, wenn die Datei im Data Dictionary beschrieben ist.

- FILE TAPEOUT NOREWIND
- FILE TAPEOUT NO OPEN

Die Kombination der Schlüsselworte INPUT/OUTPUT und der Batcherweiterungen ist unterstützt, wenn das Schlüsselwort DD angegeben wird.

Beispiel: FILE CPGWRK DD OUTPUT NO OPEN

Data Division (Working Storage Section) 3500

In der Data Division werden Felder und Feldgruppen definiert. Die einfachste Form ist die Definition eines Alphafeldes. Sie besteht aus dem Namen des Feldes und der Feldlänge in Bytes. Die Anweisung ...

```
-   ALPHA 7.
```

... definiert ein Feld mit dem Namen 'ALPHA' in der Länge von 7 Bytes als alphanumerisches Feld. Es können mehrere Felder in einer Zeile definiert werden.

```
- ANTON 5. BERTA 10. CHARLY 25. DORA 7.
```

Feldnamen, die kleiner oder gleich sechs Stellen lang sind, werden unverändert in das generierte CPG-Programm übernommen. Bei der formatfreien Programmierung dürfen Feldnamen jedoch auch größer als sechs Stellen sein, und zwar maximal 30 Stellen. Diese Felder werden mit einem automatisch erstellten Namen, bestehend aus der Konstanten 'CPG' und einer dreistelligen Nummer zwischen 001 und 999 , ins generierte Programm übernommen.

```
-    HEINRICH 27.          ergibt ein Feld 'CPG001' mit der Länge 27.
```

Bei der Definition von numerischen Feldern wird die Anzahl der Dezimalstellen durch ein Blank getrennt an die Feldlänge angehängt.

```
-    WERT 7 2.
```

definiert ein numerisches Feld mit dem Namen 'WERT' mit 7 Stellen, davon 2 Dezimalstellen.

Feldgruppen werden in der Form Name, Anzahl Elemente, '*', Länge, Dezimalstellen, definiert.

```
-  FG 10 * 5.
```

definiert eine alphanumerische Feldgruppe mit 10 Elementen von jeweils 5 Bytes Länge mit dem Namen 'FG'.

```
-  FN 10 * 5 2.
```

definiert eine numerische Feldgruppe FN mit 10 Elementen. Jedes Element ist 5 Stellen groß und hat zwei Dezimalstellen.

TWA-Überlagerung 3510

Es können auch Überlagerungsfelder (Overlay - Felder) definiert werden. Ein Überlagerungsfeld ist ein Feld, das in weitere Felder unterteilt ist. Es wird wie eine Feldgruppe definiert, als Anzahl der Elemente wird aber 0 (Null) angegeben.

Überlagerungsfelder können numerisch oder alphanumerisch sein.

Auf jede Spezifikation eines Überlagerungsfeldes müssen die Defini-

tionen der Felder folgen, in die der Bereich unterteilt werden soll. Diese werden als gewöhnliche Felder oder Feldgruppen definiert.

Ein Überlagerungsfeld kann auch weitere Überlagerungen enthalten.

Es ist darauf zu achten,dass genügend Felder definiert werden, um den gesamten für ein Überlagerungsfeld definierten Bereich zu füllen. Als Kontrolle kann das Blockdiagramm herangezogen werden.

Beispiel:

```
- ANSCHR   0 * 60
- NAME         20
- ORT          20
- STR          20
```

Die drei Felder NAME, ORT, STR können infolge der Überlagerung auch unter dem gemeinsamen Namen 'ANSCHR' angesprochen werden; beim Einlesen des Feldes ANSCHR werden auch die Felder NAME, ORT und STR gefüllt.

Bei numerischen Feldern ist bei Überlagerung zu beachten, dass diese intern in gepacktem Format gespeichert sind.

Überlagerung mit ORG 3520

Eine andere Möglichkeit der TWA-Überlagerung bietet das Schlüsselwort ORG. In Verbindung mit einem Feldnamen setzt ORG in der TWA auf der Stelle wieder auf, auf der das angegebene Feld beginnt. Wird ORG ohne Feldnamen gesetzt, so bewirkt dies ein Aufsetzen hinter dem höchsten bisher beschriebenen Byte der TWA, also auf die nächste 'freie' Stelle.

Beispiel:

```
- -D.     F1  10.     * Byte  1 - 10
-         F2  10.     * Byte 11 - 20
-    ORG  F1.         * Positionieren auf Byte 1
-         F3   5.     * Byte  1 -  5
-         F4   5.     * Byte  6 - 10
-    ORG.             * Positionieren auf Byte 21
-         F5   3.     * Byte 21 - 23
```

Um Redefinitionen noch flexibler gestalten zu können, kann jetzt auch das Ende einer Redefinition mit ORG-END explizit angegeben werden.Nach dem Schlüsselwort ORG-END muss ein bereits definierter Feldname stehen.

Beispiel:

```
DEFINE STRUKT1
ORG SATZ
  FELD1  10
  FELD2  20
  ORG FELD2
    FELD3 2
    ORG-END FELD2
  FELD4  15
```

Data Dictionary in der Data Division 3530

Strukturen, die im Data Dictionary beschrieben sind, können mit DEFINE 'Strukturname' in die Data Division übernommen werden.

Soll eine Satzart der Struktur angezogen werden, so wird diese zweistellig an den DEFINE-Befehl angehängt (siehe Beispiel).

Optional kann für die Satzart wie in den übrigen Divisions das Schlüsselwort TYPE angegeben werden.

In diesen Fällen werden die Feldbeschreibungen des Data Dictionary zur Umwandlungszeit in die Data Division übernommen.

Beispiele:

```
- -D.  DEFINE KUNDEN
-      DEFINE CPGWRK 80
-      DEFINE ARTSTA TYPE 01
```

Wenn das gleiche Feld in der Data Division mehrfach definiert wird, kommt es zu der Fehlermeldung ' ...doppelt definiert '. Arbeitet man in der Data Division mit Define Dictionary - Struktur, dann tritt dieser Fall auf, wenn ein Feld in mehreren Strukturen liegt. Durch Define Multiple ist es möglich, ein Feld mehrfach zu definieren.

Beachte:

Diese mehrfach definierten Felder werden intern auf Kommentar gesetzt, dürfen also nicht Teil einer Redefinition (Überlagerung) sein.

Wenn nur Feldnamen eingetragen werden, so wird die Felddefinition dem QDDS entnommen. Lange Feldnamen sind in Verbindung mit DDL in der Options-Anweisung unterstützt.

Außerdem wird ein FILLER in der Data Division (Define DD) generiert. (Nur bei Dateien mit Directory Field Check).

Beispiel:

```
DEFINE DEMO
  F1              10    * FELD1
   CPGFIL         10    * CPGFILLER
  F2              20    * FELD2
  F3              15    * FELD3
   CPGFIL         65    * CPGFILLER
  F4              20    * FELD4
```

Externe Felder (Schlüsselwort EXT) 3540

Das Schlüsselwort EXTern im Anschluss an die Feldbeschreibung kennzeichnet ein Feld als extern. Das bedeutet, dass das Feld nicht in die TWA des Programms übernommen wird.

Der Programmierer kann somit auf externe Felder zum Beispiel in der CSA zugreifen, wobei er jedoch sicherstellen muss, dass das angegebene Feld auch unter gleichem Namen und mit gleichen Eigenschaften außerhalb der TWA vorhanden ist.

Ausrichtung auf Wortgrenzen 3550

Alphafelder können bei der Definition in der TWA ausgerichtet werden; dazu stehen folgende Schlüsselwörter zur Verfügung:

Double für die Ausrichtung auf Doppelwortgrenze,
Full für die Ausrichtung auf Vollwortgrenze und
Half für die Ausrichtung auf Halbwortgrenze.

Das Schlüsselwort wird an die Definition angehangen.

Es ist nur ein Schlüsselwort im Anschluss an eine Definition möglich; die Kombination der Ausrichtung auf Wortgrenzen mit dem Schlüsselwort EXT für externe Felder ist nicht unterstützt.

Forms-Division (Druckersteuerung) 3600

Für Online-Drucker, die im Line Mode (mit Zeilentransport und Kanalvorschub) arbeiten, muss am Ende der Data Division eine Formularbeschreibung gegeben werden.

Die Formularbeschreibung wird eingeleitet durch den Parameter FORMS. Danach folgt der Name Druckers aus der Files Division.

Die Formularlänge wird standardmäßig mit 72 Zeilen (12 Zoll) angenommen. Wird eine andere Formularlänge gewünscht, ist das Schlüsselwort LENGTH gefolgt von der gewünschten Zeilenzahl anzugeben.

Kanal 1 wird standardmäßig auf Zeile 6 und Kanal 12 auf Zeile 69 angenommen. Bei anderen Vorschüben sind alle Kanalvorschübe paarweise in der Reihenfolge 'Zeile', 'Kanal', mit oder ohne Schlüsselworte anzugeben. Die Formularbeschreibung darf nicht über eine Zeile hinausgehen. Die Schlüsselworte haben hier nur beschreibenden Charakter und können vom Programmierer frei gewählt werden.

Beispiele:

- FORMS DR01. * Drucker DR01, 72 Zeilen lang, Zeile 6 Kanal 1, 69 12
- FORMS DR02 LENGTH 36. * Drucker DR02 36 Zeilen, Z 6 Kan 1, Z 69 K 12
- FORMS DR03 LINE 3 CHAN 01 LINE 12 CHAN 02 LINE 66 CHAN 12. *Lge. 72
- FORMS DR04 LENGTH 72 LINE 1 CHANNEL 1 LINE 69 CHANNEL 12.
- FORMS DR05 LENGTH 36 003 01 066 12.
- FORMS DR06 LENGTH 36 ZEILE 3 KANAL 1 ZEILE 66 KANAL 12.
- FORMS DR07 LENGTH 36 L 3 C 01 L 66 C 12.

Wird keine Formularbeschreibung angegeben, so werden intern die Defaultwerte angenommen. Die Forms-Division ist somit optional.

Input Division (Grammatik) 3700

In der Input Division werden die Eingabe-Strukturen der für das Programm erforderlichen Daten beschrieben. Wir unterscheiden dabei zwei Arten von Eingabebeschreibungen: Die Satzbeschreibung und die Feldbeschreibung. Für jede Datei, von der Daten gelesen werden, ist eine Eingabebestimmung erforderlich. Eine Eingabebestimmung besteht aus einer Satzbestimmung, die zwingend als erstes Statement einer Eingabebestimmung verlangt wird, und Feldbestimmungen, die in der Regel beim Umwandeln vom Data Dictionary in des Programm eingefügt werden.

Satzbeschreibung 3710

Es wird unterschieden nach Datei- und Feldbeschreibungen:

Dateien	KW DN (VAR) (BA)
Felder	KW DN (TP)

KW = Schlüsselwort
DN = Datei-, Segment- oder Feld(gruppen)-Name
VAR = variable Eingabepositionen
BA = Bedingungsabfrage
TP = Typ, Name einer bestimmten Feldauswahl

Schlüsselwort (KW).

FILE bedeutet, dass die Daten aus einer Datei gelesen werden. Eine Datei ist jede Art von externem Speicher, aus dem Daten gelesen werden können, z.B. Platte, Bildschirm, Storage usw. Mit FILE können jedoch auch Daten aus logischen Dateien, z.B. Datasets, HL1-Bausteinen, Datenbanken usw. gelesen werden.

FIELD bedeutet, dass die Daten aus einem Feld oder einer Feldgruppe im Hauptspeicher gelesen werden, das oder die vorher in der Data Division definiert werden muss.

SEGM Segmente sind Strukturen, die zu VSAM-Dateien oder HL1-Datasets gehören und mit READI eingelesen werden. Es gelten die gleichen Syntaxregeln wie bei FILE. SEGMente werden immer direkt im Anschluss an die FILE-Bestimmung codiert, zu der sie gehören.

Dateiname (DN)

Nach dem Schlüsselwort folgt zwingend der Name der Datei oder des Feldes oder der Feldgruppe. Je nach Schlüsselwort gelten für die Namen unterschiedliche Regeln.

FILE der Name der Datei darf maximal acht Stellen lang sein. Eine
SEGM Ausnahme gilt für Namen von Datenstrukturen, die auf sechs

Stellen begrenzt sind.

FIELD Feldnamen dürfen bis zu dreissig Stellen lang sein. Zu beachten sind jedoch die in Abschnitt 2 (Felder, Feldgruppen) aufgeführten Einschränkungen, vor allem für Feldgruppen, die indiziert verarbeitet werden sollen.

Ansteuern von Feld-Selektionen

Sind bei der Feldselektion (Schlüsselwort FIELD) verschiedene Möglichkeiten gegeben, so kann zur Auswahl einer bestimmten Input-Beschreibung ein Name in der Satzbestimmung vergeben werden. Dieser Name wird in Verbindung mit dem Schlüsselwort TYP an das Statement angehängt.

Die Feldselektion kann auch dem Data Dictionary entnommen werden. Das Schlüsselwort TYPE kann deshalb verschiedene Bedeutungen haben: Es kann für die Satzart des Data Dictionary stehen, aber auch für den Namen der Feld-Selektion. Aus Gründen der Eindeutigkeit kann für den Namen der Feld-Selektion auch das Schlüsselwort SELECT-type angegeben werden.

In der Input - Satzbestimmung kann zusätzlich zu den Data Dictionary Schlüsselwörtern noch DEFine angegeben werden. Damit wird sichergestellt, dass alle Felder der Struktur für das Programm definiert werden.

```
Beispiel: -I.
            FILE DATEI DD DEF.

oder    : -I.
            FILE DATEI KF 01 1 C A.
            FILE       DD REF DATEI TYPE XX DEF.
```

Variable Eingabepositionen (VAR)

Eine besondere Verarbeitungsart ist die Arbeit mit variablen Eingabepositionen. Soll eine Datei auf diese Art verarbeitet werden, so muss unmittelbar im Anschluss an den Dateinamen das Schlüsselwort VAR angegeben werden.

Diese Verarbeitungsart ist derzeit eine Ausnahme; sie wird beispielsweise benutzt, um Eingaben bei Satzlängen von mehr als 8000 Bytes bearbeiten zu können.

Der Verschiebungsfaktor steht dabei im internen Feld CPGFIS (siehe Beispiel im Kapitel 8000).

Bedingungsabfrage (BA).

Die Bedingungsabfrage ist für die einzelnen Dateitypen unterschiedlich. Die allgemeine Form richtet sich daher nach dem Dateityp:

Platte, Band, Reader	(TP) (BD) (CA) (AND) (CA) (AND) (CA) mit CA = PS (NOT) CD CH

Dataset	(TP) (BD) (CA) (AND) (CA) (AND) (CA)
Bildschirm	(TP) (BD)
Data Dictionary	TP (KW SA) (KW REF)
Datenstruktur	TP (LG)
HL1 Datenkanal	TP
Feld	(KW) TP

In dieser Darstellung bedeuten:

```
TP  =  Dateityp / Feldaufbereitungstyp
BD  =  Bedingung (Schalter 01 bis 99)
SA  =  Satzart (über Schlüsselwort 'TYPE')
REF =  Referenzstruktur (über Schlüsselwort 'REF')
CA  =  Zeichen Abfrage
 PS =  Position im Satz
 CD =  Zeichen Code (Character Digit Zone)
 CH =  Zeichen
LG  =  Länge der Datenstruktur
AND =  Verknüpfung mehrerer Abfragen
NOT =  Umkehrung (Bedingung ist nicht erfüllt)
```

Dateityp / Feldaufbereitungstyp (TP)

Der Dateityp ist in der Regel optional.

Dateien beliebige Eintragung (z.B. KF = keine Folge)

In folgenden Spezialfällen ist die Angabe des Typs vorgeschrieben:

Data Dictionary	DD
Datenstruktur	DS
DL1 Datenbank	DS (Eingabe über Datenstruktur)
HL1 Datenkanal	HS
Feldaufbereitung	beliebiger Name, bis zu 30 Stellen

Bedingung (BD)

Hier wird ein Schalter zwischen 01 und 99 eingesetzt. Bei Bedingungsabfrage wird der Schalter dann gesetzt, wenn alle abgefragten Bedingungen erfüllt sind. Eine Bedingung ist zwingend erforderlich, wenn eine Zeichenabfrage (CA) folgt.

Beispiel dazu: ----------

```
- FILE PLATTE KF  99  1 C 0.  100 101 FELD2A
- FILE PLATTE     98  2 C 0.  100 101 FELD2B
```

Ist in diesem Fall das erste Zeichen der Datei gleich 0, so wird von der Datei Platte in das FELD2A eingelesen, aber nicht in das FELD2B. Wenn eine Bedingung erfüllt und ein Schalter gesetzt ist, werden die folgenden Satzbestimmungen der gleichen Datei nicht mehr verarbeitet.

Abhilfe:

```
- FILE PLATTE KF   #  1 C 0.  100 101 FELD2A
- FILE PLATTE KF   #  2 C 0.  100 101 FELD2B
```

In diesem Fall wird auf den Eingabeschalter verzichtet. Ein Nummernzeichen muss eingetragen werden. Jetzt sind die Eingabe-Satzbestimmungen voneinander unabhängig, so dass sowohl FELD2A als auch FELD2B eingelesen werden können.

Zeichenabfrage (CA).

Eine Zeichenabfrage bezieht sich immer auf eine Speicherstelle in einem Datensatz, die auf einen bestimmten Wert abgefragt wird. Die Abfrage kann bis zu drei mal für unterschiedliche Zeichen wiederholt werden. Abgefragt wird, ob auf einer bestimmten Position (PS) ein bestimmtes Zeichen (CH) vom Typ (CD) steht oder nicht (NOT) steht.

PS ist der bis zu vierstellige Wert der Position relativ zum Satzanfang. PS = 1 für die erste Position im Satz.

CD gibt an, wie geprüft werden soll:

C	oder	CHARACTER	das ganze Zeichen wird geprüft.
D	oder	DIGIT	nur der Ziffernteil wird geprüft.
Z	oder	ZONE	nur der Zonenteil wird geprüft.

CH gibt an, auf welchen Wert geprüft werden soll. Z. B. 'A': Enthält die zu prüfende Stelle das Zeichen 'A' ? (# für Blank ?)

NOT kehrt die Abfrage um, das heisst enthält die zu prüfende Stelle irgend ein anderes Zeichen als 'CH'.

Bei einer Zeichenabfrage sind die Parameter TP und BD zwingend erforderlich.

Beispiele:

2 C A	enthält Stelle 2 das Zeichen 'A'
1234 D 7	enthält Stelle 1234 die Ziffer sieben
1 NOT C 0	enthält Stelle eins nicht die Ziffer Null

Länge der Datenstruktur (LG).

Für die Datenstruktur kann hier die volle Länge der Datenstruktur eingetragen werden, wenn die Struktur in der Eingabebestimmung nur zum Teil beschrieben wird.

Beispiele für Satzbestimmungen:

- FILE BILD
- FILE PLATTE VAR
- FILE KUNDEN DD
- FILE KUNDEN VARIABEL DD 99 11 CHAR *.
- FILE PLATTE KF 11 1 NOT CHAR A 2 CHAR B 3 CHAR #.
- FILE ARTIKEL AA 01 1 C A AND 2 C R.
- FILE HUGO DS 200.
- FILE XKANAL HS.

Input-Satzbestimmungen in Verbindung mit Data Dictionary sind in Kapitel 2600 ausführlicher beschrieben.

- FIELD HUGO.
- FIELD AF TYPE SATZ1
- FIELD ALPHANUMERIC-STRING.

Nicht erlaubt sind:

- FILE LANGER-HEINRICH. * Dateiname größer als 8 Stellen
- FILE ARTIKEL 1 C A AND 2 C R. * Dateityp und Bedingung fehlt
- FIELD HUGO AA 01. * Dateityp und Bedingung nicht erlaubt

Feldbeschreibung 3750

Die allgemeine Darstellungsform einer Feldbestimmung lautet:

```
-----------------------------------------------------------------------
            (SF) VP BP (DP) FN (SP) (BD) (BD) (BD)
-----------------------------------------------------------------------
```

SF = Speicherungsform (packed, binary, logical)
VP = von Position im Satz
BP = bis Position im Satz
DP = Anzahl Dezimalstellen
FN = Feldname
SP = Schlüsselwort Lichtstiftbezugszahl
BD = Schalter 01 - 99 oder L0 - L9

Speicherungsform (SF)

Für alphanumerische Datenfelder entfällt diese Eintragung. Für numerische Felder wird hier eine Eintragung vorgenommen, wenn die Daten in einem der platzsparenden gepackten Formate gespeichert werden.

P	oder	PACked	das Feld ist in gepacktem Format gespeichert.
B	oder	BINary	das Feld ist in binärem Format gespeichert.
L	oder	LOGical	das Feld ist logisch gepackt gespeichert.
O	oder	OPTimize	das Feld ist in gepacktem Format mit gerader Stellenzahl gespeichert.

Von Position (VP)

Hier wird die Position eingetragen, bei der das Feld beginnt. Die Eintragung ist numerisch, bis zu vier Stellen lang und zwingend vorgeschrieben.

Bis Position (BP)

Hier wird die Position eingetragen, bei der das Feld endet. Die Eintragung ist numerisch, bis zu vier Stellen lang und zwingend vorgeschrieben.

Dezimalstellen (DP)

Eine Eintragung in dieser Position kennzeichnet das Feld als numerisch. Bei numerischen Feldern wird hier die Anzahl Dezimalstellen eingetragen. Die Eintragung ist numerisch und darf nicht größer als eine Stelle sein.

Name des Feldes oder der Feldgruppe (FN).

In dieser Position muss zwingend ein gültiger Feldname eingetragen werden (siehe auch Abschnitt 2, Felder).

Schlüsselwort Lichtstiftbezugszahl

SP steht für Selector Pen und kennzeichnet eine Bezugszahl als Lichtstiftbezugszahl (nur für Bildschirmdateien).

Beispiel: - 0205 0279 PAGE SP 20

Bezugszahlen (BD)

Es können drei Eingabebezugszahlen angegeben werden, die bei der Eingabe gesetzt werden, wenn das Eingabefeld größer (erste BD), kleiner (zweite BD) oder gleich Null bzw. Blank (dritte BD) ist. Die Stellen, die keine Bedingung enthalten, müssen mit Nummernzeichen besetzt werden.

Es kann ein Gruppenschalter L0 - L9 gesetzt werden.

Die drei Möglichkeiten Gruppenschalter, Eingabebezugszahlen und Lichtstiftbezugszahl schließen sich gegenseitig aus.

Beispiele für Feldbestimmungen: ----------

```
-    1 2 SA.           * von 1 bis 2 steht die Satzart.
-    3 9 2 WERT.       * von 3 bis 9 steht der Wert mit 2 Dezimalst.
-    P 10 11 0 LNR.    * von 10 bis 11 steht die laufende Nr gepackt.
-    1 2 SA. 3 9 2 WERT. P 10 11 0 LNR.      * oder so.
-   0202 0279 PAGE  SP 01.  *  Lichtstiftbezugszahl (ab) 01
-     11 12 WGRP  L2.       *  Gruppenwechsel L2 beim Feld WGRP
-  PAC 1  3 2  NUM  # 10.   *  Schalter 10, wenn NUM kleiner 0 ist
```

Beispiele für Eingabebeschreibungen: 3790

```
- FILE KUNDEN DD.    * Alle Feldbeschreibungen werden dem Data Dictio-
-                    * nary entnommen. Die Eingabebeschreibung für die
-                    * Datei KUNDEN ist damit komplett.

- FILE KONTEN.       * manuelle Beschreibung
-             1   5    KONTO
-             6  30    TEXT
-        P   31  35 0  UMS.     * Feldgruppe 12 Felder von 31 bis 90
-            91  92 0  MONAT

- FILE KONTEN.
-      1 5             KONTO
-      6 30            TEXT
-      PACKED 31 35 0  UMS
-      91 92 0         MONAT

- FILE KONTEN. 1 5 KONTO. 6 30 TEXT. P 31 35 0 UMS. 91 92 0 MONAT.
```

Operationen in der Procedure Division 3800

Die folgenden Standard-Operationen stehen in der Procedure Division für alle Benutzeroberflächen zur Verfügung.

ADD	Addieren
AFOOT	Mittelwert einer Feldgruppe
BEGAS	Umschalten auf assembler-codierte Programmteile
BEGSR	Unterprogramm starten
BITON	Bit ansetzen
BITOF	Bit aussetzen
BREAK	Schleife beenden
CAB	Vergleichen und verzweigen
CALL	Unterprogramm (andere Programmiersprache) aufrufen
CALLD	Dataset aufrufen
CALLM	Methodenbank Routine aufrufen
CAS	Gruppe vergleichen und Unterprogramm aufrufen
CBS	Vergleichen und Unterprogramm aufrufen
CHAIN	Datensatz direkt lesen
CHECK	Datensatz auf Vorhandensein prüfen
CHANG	Felder austauschen
CLEAR	Alle Felder auf Blank löschen
CLOSE	Datei schließen
CLRIN	Schalter abhängig von einem Index löschen
COMP	Vergleichen
COMRG	Verständigungsbereich aufrufen
CONT	Schleife unterbrechen
CONVT	Konvertieren eines alphanumerischen Feldes
COPY	Assembler Copy Book einfügen
DEBUG	Testhilfe oder QDF ermöglichen/unterdrücken
DELC	Zeichen entfernen
DELET	Datensatz löschen
DEQ	Warteschlange freigeben
DIV	Dividieren
DLI	DL 1 Aufruf
DO	Programmblock verarbeiten
DSPLY	Konsolmeldung ausgeben
DUMP	Speicherauszug drucken
EDIT	Alphafeld aufbereiten
ELIM	Zeichen eliminieren
ELSE	Andernfalls
END	Ende eines Programmblocks
ENDAS	Ende eines assembler-codierten Programmteils
ENDDO	Ende eines DO Blocks
ENDEV	ENDE eines EVALUATE-Blocks
ENDIF	Ende eines IF Blocks
ENDSR	Ende eines Unterprogramms
ENQ	In Warteschlange einreihen
EREAD	Bildschirm Aus-/Eingabe
EVALUATE	Mehrfachalternative
EXCPT	Ausgabe
EXHM	HL1-Baustein ausführen
EXIT	Anderes Programm aufrufen
EXITD	Anderes Programm aufrufen mit Datenübergabe
EXITI	Anderes Programm aufrufen über Intervall control
EXITP	Anderes Programm aufrufen (wie EXIT)
EXITS	Programm (später) an anderes Terminal schicken
EXITT	Anderes Programm aufrufen mit Bildschirmdaten
EXPR	Anderes Programm ausführen

EXSR	Unterprogramm ausführen
FILL	Alphafeld mit Zeichen füllen
FIND	Durchsuchen einer Datenview
GETHS	HL1-Datenkanal erneut übertragen
GETIN	Schalter der höheren Stufe abfragen (nur HL1)
GETMI	Schalter der höchsten Stufe abfragen (nur HL1)
GOTO	Verzweigen
IF	IF Block
INDOF	Schalter von bis löschen
INDON	Schalter von bis setzen
JLB	Alphafeld linksbündig verschieben hinten Blanks
JRB	Alphafeld rechtsbündig verschieben vorn Blanks
JRC	Alphafeld rechtsbündig verschieben vorn Zeichen
JRZ	Alphafeld rechtsbündig verschieben vorn Nullen
LIST	Ausgabe programmextern beschriebener Listen
LOADT	Geretteten Bildschirm zurückholen
LOKUP	Schlüssel in der Feldgruppe aufsuchen
MACRO	Angabe eines Assembler-Befehls
MAP	QSF-Map lesen
MAPD	QSF-Map ausgeben und lesen im Dialog
MAPI	QSF-Map lesen im Dialog
MAPO	QSF-MAP auf den Bildschirm ausgeben
MAPP	QSF-Map auf einen Drucker ausgeben
MLLZO	Vorzeichen übertragen
MOVE	Feldinhalt übertragen
MOVEA	Feldgruppeninhalt übertragen
MOVEL	Feldinhalt linksbündig übertragen
MOVEN	Alphanumerisches in numerisches Feld übertragen
MOVEV	variable MOVE-Operation
MULT	Multiplizieren
MVR	Rest einer Division übertragen
OPEN	Datei eröffnen
QSSA	Qualifizierter SSA (DL1)
PARM	Parameter übergeben beim Befehl CALL
PROGRAM	QPG-Programm ausführen
PROT	Schutzcode vergeben (für CPG3-Anwender)
PURGE	Speicher freigeben
PUTIN	Schalter zur höheren Stufe übertragen
PUTMI	Schalter zur höchsten Stufe übertragen
READ	Datensatz lesen
READB	Datensatz rückwärts lesen
READI	File Work Area lesen / Bildschirmdaten übertragen
READP	Seite lesen
REPLC	Zeichen ersetzen
RNDOM	Wahlfreie Verarbeitung
ROLL	Feldgruppe rollen
ROLLB	Feldgruppe rückwärts rollen
SAVET	Bildschirminhalt retten
SCAN	Alphafeld nach einer Zeichenfolge durchsuchen
SDUMP	Speicherauszug
SELCT	Feld auslesen
SETOF	Schalter löschen
SETIN	Schalter abhängig von einem Index setzen
SETIX	Index abhängig von einem Schalter setzen
SETON	Schalter setzen
SETLL	Zeiger auf einen Schlüssel setzen
SORT	Feldgruppe sortieren
SORTA	Feldgruppe sortieren
SQRT	Wurzel ziehen
SUB	Subtrahieren
SYNCP	Syncpoint setzen

TAG	Merkmal setzen
TESTB	Bit prüfen
TESTF	Feld auf numerische Zeichen prüfen
TESTN	Feld auf numerische Zeichen prüfen
TESTT	Bilbschirmname prüfen
TIME	Zeit setzen
TWALD	Gerettete TWA von Temporary Storage laden
TWASV	TWA auf Temporary Storage speichern
UCTRN	Übersetzen in Großbuchstaben
UPDAT	Datensatz verändern
USSA	Unqualifizierter SSA (DL1)
WAIT	Warten
WHEN	Kennzeichnung einer Bedingung
WRITE	Datensatz neu anlegen
XFOOT	Summe einer Feldgruppe rechnen
Z-ADD	Löschen und addieren
Z-SUB	Löschen und subtrahieren

Grammatik 3810

Eine Operation in der Procedure Division besteht aus vier Grundelementen:

Bedingungsabfrage	ON
Ausführung der Operation	OP
Serviceabfrage	SV
Bedingungen setzen	BD

Der Einfachheit halber werden diese Elemente im nachfolgenden Text in der Kurzform dargestellt. Mit Ausnahme der Grundoperation können alle anderen Elemente wahlweise genutzt werden. Wird ein Element wahlweise unterstützt, so wird es im Text durch Klammern dargestellt (ON). Wird ein Element zwingend verlangt, so wird es ohne Klammern dargestellt, wird es nicht unterstützt, so entfällt bei der Beschreibung das Kürzel.

Eine Operation kann also in der allgemeinen Form

```
-----------------------------------------------------------------
              (ON) OP (SV) (BD)
-----------------------------------------------------------------
```

dargestellt werden. Das bedeutet: Die Elemente der Operation werden in der Reihenfolge Bedingungsabfrage (ON), Grundoperation (OP), Serviceabfrage (SV), Bedingungen (BD) beschrieben, wobei OP zwingend und ON, SV und BD wahlweise vorhanden sein müssen.

Die Bedingungsabfrage (ON).

Bei fast allen Operationen kann die Ausführung von einer oder mehreren Bedingungen abhängig gemacht werden. Wird auf die Bedingungsabfrage verzichtet, so wird die Operation in jedem Falle ausgeführt. Werden Bedingungen abgefragt, so wird die Operation nur dann ausgeführt, wenn alle Bedingungen erfüllt sind.

Bedingungen können numerische Schalter von 01 bis 99 sein, die von anderen Operationen gesetzt werden, oder Programmfunktionstasten von Bildschirmen oder andere Schalter (z.B. EF für End of File, siehe Abschnitt 2260, Schalter).

Die Bedingungsabfrage kann in der folgenden Form allgemein dargestellt werden :

```
-----------------------------------------------------------------
              KW (NOT) BD (AND) (NOT) BD (AND) (NOT) BD
-----------------------------------------------------------------
```

Die Darstellung ist für alle Operationen gleich und wird daher bei der Beschreibung der einzelnen Operationen nur noch in der Kurzform (ON) dargestellt.

Eine Bedingungsabfrage wird immer durch das Schlüsselwort 'ON' eingeleitet. Fehlt dieses Schlüsselwort und werden trotzdem Bedingungen angegeben, so kommt es zu Fehlinterpretationen des Compilers, die zum Programmabbruch führen. Das Schlüsselwort 'ON' kann auch durch das deutsche Schlüsselwort 'BEI' ersetzt werden.

Dem Schlüsselwort 'ON' können bis zu drei Bedingungen folgen, die miteinander 'UND'-verknüpft werden. Um dies im Text zu verdeutlichen, kann der Programmierer das Wort 'AND' oder 'UND' einfügen. BD steht für den immer zweistelligen Bedingungsschalter. Soll die Operation ausgeführt werden, wenn eine Bedingung nicht erfüllt ist, so wird dies durch das Schlüsselwort 'NOT' oder 'NICHT' vor der Bedingung zum Ausdruck gebracht.

Beispiele für Bedingungsabfragen:

ON 11	bei Schalter 11
ON P1	bei Programmfunktionstaste 1
ON NOT 12	nicht bei Schalter 12
ON 11 12	bei Schalter 11 und 12
ON 11 21 31	bei Schalter 11,21 und 31
BEI 11 12 UND 13	bei Schalter 11,12 und 13
ON 11 12 AND NOT 13	bei Schalter 11,12 und nicht 13
ON NOT PB AND NOT EF AND NOT 10	bei nicht PF11,nicht EF,nicht 10

Die Operation (OP).

Für die Operationen gelten je nach Art der Operation verschiedene grammatikalische Regeln, die bei der Beschreibung der einzelnen Operationen im einzelnen erläutert werden. Dieser Abschnitt beschränkt sich daher auf die generellen Elemente der Operation.

Eine Operation besteht aus den folgenden Grundelementen:

Operations-Schlüsselwort	OC (Op.-Code)
erweiterter OP.-Code	OE
Faktor 1	F1
Faktor 2	F2
Ergebnis	EG
Dummy-Worte	DY
Operatoren	OK

Die allgemeine Darstellung der Grundoperation lautet wie folgt:

```
-----------------------------------------------------------------------
             OC (OE) (DY) (F1) (DY) (NOT) (OK) (F2) (DY) (EG)
oder kurz:   OC (F1) (F2) (EG)
-----------------------------------------------------------------------
```

Operanden (F1 F2 EG)

Die Elemente F1, F2 und EG werden auch Operanden genannt. Eine Operation braucht in jedem Falle den Operations-Code, aber nicht immer alle Operanden. Auch die Reihenfolge der Operanden kann je nach Operation oder Benutzer-Oberfläche unterschiedlich sein. So kann beispielsweise bei verschiedenen Operationen EG oder F1 vor OC stehen.

```
A MULT B C                      F1 OP F2 EG
MULTIPLY A BY B GIVING C        OP F1 DY F2 DY EG
C = A * B                       EG OP F1 OK F2
```

Operatoren (OK)

Bei Vergleichsoperationen kann zwischen F1 und F2 ein Operator stehen, der gegebenenfalls durch verbindende Textworte ergaenzt wird. Z.B.:

```
IF A = B                        OP F1 OK F2
IF A EQ B                       OP F1 OK F2
IF A IS EQUAL TO B              OP F1 DY OK DY F2
IF A IS NOT GREATER THAN B      OP F1 DY OK OK DY F2
DO UNTIL A IS GREATER THAN B    OP OE F1 DY OK DY F2
DO WHILE A = B                  OP OE F1 OK F2
```

Gültige Darstellungsformen für Operatoren sind:

```
>    oder   GT   oder   (IS) GREATER (THAN)      größer als
<    oder   LT   oder   (IS) LESS (THAN)         kleiner als
=    oder   EQ   oder   (IS) EQUAL (TO)          gleich
>=   oder   GE   oder   (IS) NOT LESS (THAN)     größer gleich
<=   oder   LE   oder   (IS) NOT GREATER (THAN)  kleiner gleich
><   oder   NE   oder   (IS) NOT EQUAL (TO)      ungleich
<>                                               ungleich
```

Beispiele für Operationen:

```
   EXCPT                                OP
   DO 10 TIMES                          OP F2 DY
   DO FROM X TO Y                       OP DY F1 DY F2
   MOVE TEXT TO LINE                    OP F1 DY F2
   X = 0                                EG OP F2
   X = A + B                            EG OP F1 OK F2
   READ OTTO                            OP F2

   MAP ARTIKEL                          OP F2
   OBEN                                 F1
   OBEN TAG                             F1 OP
```

Serviceabfrage.

Die Serviceabfrage besteht, soweit bei der einzelnen Operation unterstützt, aus einem einzigen Schlüsselwort. Für Service-Schlüsselworte bestehen keine Einschränkungen, das heisst sie können auch als Namen für Variable verwendet werden. Gültige Schlüsselworte sind:

A	AFTer	ARRay	BEFore	BLAnk	C	CHAracter
CHECk	CLEar	D	E	EXTern	F	H
HEX	I	INDex	INPut	K	L	LEFt
LOOP	LOW	N	O	P	R	RIGht
ROLl	ROUnded	RUNden	S	SAVe	SEConds	T
TIMe	U	UPDate	V	VARiable	1	

Es genügt, jeweils die hier groß geschriebenen Buchstaben (eine oder drei Stellen) anzugeben. Im Zweifelsfall wird die erste Stelle des erste Stelle als Schlüsselwort akzeptiert. Eventuelle Fehler bemerkt somit der CPG-Compiler.

Output - Division 3900

In der Output-Division wird beschrieben, wie und wohin die Datenfelder ausgegeben werden. Wir unterscheiden dabei 'Satzbestimmungen' und 'Feldbestimmungen', je nachdem ob einheiten- oder feld-spezifische Angaben gemacht werden.

Data Dictionary in der Output Division 3905

Durch Einsatz des Data Dictionary in der Satzbestimmung erreicht man, dass keine Feldbestimmungen codiert werden müssen. Data Dictionary in der Output Division ist auch beschrieben in Kapitel 2600.

Grundsätzlich gilt als Syntaxregel, dass die vollständigen Data Dictionary-Parameter direkt im Anschluss an den Dateinamen angegeben werden. Die übrigen Schlüsselworte werden nach den unten angegebenen Regeln daran angehängt.

Satzbestimmung für EXCPT-Ausgaben (Dateien aller Art) 3910

Das erste Statement einer Ausgabebestimmung ist immmer die Satzbestimmung, das heisst, die Angabe der Datei, zu der die Datenfelder übertragen werden.

Das Schlüsselwort 'FILE' bildet immer den Anfang einer solchen Ausgabebestimmung. Dem Schlüsselwort folgt unmittelbar der Name der Datei.

Je nach Art der Ein-/ Ausgabe-Einheit können dem Dateinamen unterschiedliche Schlüsselworte folgen. Es genügen jeweils die drei ersten Buchstaben und die beschriebene Reihenfolge muss eingehalten werden.

Platte

Das Schlüsselwort 'ADD' gibt bei Plattendateien an, dass ein Satz hinzugefügt und 'DEL', dass ein Satz gelöscht wird. 'ALG' bedeutet, dass bei variabler Satzlänge die Länge verändert wird.

```
-   FILE PLATTE DD          ADD
-   FILE PLATTE DD TYPE LX  ALG
-   FILE PLATTE             DEL.      * Keine Feldbestimmungen bei DEL
```

Für Dateien mit variabler Satzlänge besteht die Möglichkeit, über die Satzbestimmung der Output Division zu bestimmen, wie lang ein ausgegebener Satz sein soll. Mit den Schlüsselwörtern ADD-VAR, ALG-VAR oder mit einem VAR im Anschluss an ADD oder ALG erreicht man, dass die tatsächliche Länge des Ausgabesatzes durch das CPG-interne Feld CPGVRL bestimmt wird.

```
-   FILE PLATTE    ADD VAR  FALL1

-   FILE PLATTE DD ADD-VAR  FALL2
```

Temporary Storage

Die im folgenden beschriebenen Einträge 'I' und 'A' sollten nur in speziellen Ausnahmefällen codiert werden. Die beiden Verarbeitungsarten 'Independent' und 'Auxiliary' gibt man in der Regel bereits in der Dateibeschreibung (FILES DIVISION und Data Dictionary) vor.

Der Schlüssel 'I' gibt an, dass die Daten des Storage-Bereichs von allen Terminals gelesen werden können und im Hauptspeicher gespeichert werden. Der Eintrag 'A' bewirkt, dass die ausgegebenen Daten nur vom eigenen Bildschirm wieder eingelesen werden können und im Hilfsspeicher gespeichert werden.

Wird keine dieser beiden Angaben gemacht, so werden die ausgegebenen Daten im Hauptspeicher gespeichert und können nur vom ausgebenden Terminal wieder eingelesen werden.

Bänder

Bei der Ausgabe auf Banddateien werden keine Zusätze wie ADD, DEL und ALG angegeben.

Für Bänder mit variabler Satzlänge besteht die Möglichkeit, über die Satzbestimmung der Output Division zu bestimmen, wie lang ein ausgegebener Satz sein soll. Mit dem Schlüsselwort VAR erreicht man, dass die tatsächliche Länge des Ausgabesatzes durch das CPG-interne Feld CPGVRL bestimmt wird.

- FILE TAPOUT VARIABEL.

Drucker.

1. Für Drucker-Dateien werden Zeilentransporte durch SPAce angegeben. Dem Schlüsselwort müssen beide Angaben (für Zeilentransport vor und nach dem Drucken) folgen.

   - FILE DRUC SPACE 2 1
   - FILE PRNT SPACE # 3

2. Vorschübe werden durch das Schlüsselwort 'SKIP' angegeben, dem eine zweistellige Konstante für den Kanal folgt.

   - FILE DRUC SKIP 01 ON 11

   Der Kanal (im Beispiel 01) muss in der Forms Division beschrieben sein, wenn das Programm online ausgeführt wird oder wenn das Programm im Batch ausgeführt wird und der Drucker keine SYS-Nummer enthält.

Bildschirm.

(veraltete Verarbeitungsart !)
(entfällt bei Einsatz von QSF)

1.	UNFormatierte Ausgabe	siehe unten.
2.	ERASE	Bildschirm vor der Ausgabe löschen
3.	U oder UNProtected	Löschen aller ungeschützten Felder
	H oder BEEP	Hupe
	M oder MODifiziert	Modifizierte Felder nicht freigeben
	K oder L, N, O, S =	Kombinationen gemäß Tabelle 7030

Unformatierte Ausgaben geben die Ausgabeposition relativ zum Bildschirmanfang an. Zeile 2 Spalte 40 wird somit z.B. zur Position 120.

Unformatierte Ausgaben werden mit SELECT CPGTIO wieder eingelesen.

In der Files Division muss für einen unformatierten Bildschirm eine Satzlänge angegeben werden, die mindestens so groß ist wie die maximale Ausgabeposition; beim Bildschirm, der 24 Zeilen und 80 Spalten hat, also z.B. die Satzlänge 1920.

Eine Besonderheit ist weiterhin, dass der Programmierer bei unformatierter Ausgabe ohne Erase (auf einen unformatierten Bildschirm) für den Aufbau dieser Ausgabe selbst verantwortlich ist. An allen nicht beschriebenen Stellen werden Blanks ausgegeben.

Bedingungen

Soll die Ausgabe nur unter einer oder mehreren Bedingungen ausgegeben werden,so folgt die Anweisung 'ON' gefolgt von bis zu drei Bedingungsschaltern.

- FILE BILD ERASE ON 01.
- FILE PLATTE ON 01 AND 02 AND NOT 03.

Den Bedingungen kann auch ein bis zu 6-stelliger Name folgen, der aber auch anstelle der Schalter angegeben werden kann. Dieser Name arbeitet genau so wie die Bezugszahlen.

Satzbestimmung bei Feldaufbereitungen 3915

Feldaufbereitungen werden in der Output-Division mit Satzbestimmung und Feldbestimmung(en) beschrieben.

Datenfelder können im Speicher aufbereitet werden. Dazu wird für das Feld immer das Schlüsselwort FIELD verwendet. Feldaufbereitungen werden nicht bei der Ausgabe mit dem Befehl EXCPT verarbeitet, sondern nur über den Befehl EDIT.

Die Syntax ist außerdem unterschiedlich zur Dateiausgabe: Will man unterschiedliche Ausgabemöglichkeiten in ein Feld beschreiben, so arbeitet man sowohl in der Procedure Division als auch in der Output Division mit einem EDIT-Namen, der mit dem Schlüsselwort TYP angegeben wird.

Die Feldaufbereitung kann dem Data Dictionary entnommen werden. Das Schlüsselwort TYPE kann deshalb verschiedene Bedeutungen haben: Es kann für die Satzart des Data Dictionary stehen, aber auch für den Namen der Feldaufbereitung. Aus Gründen der Eindeutigkeit kann für den Namen der Feldaufbereitung auch das Schlüsselwort SELECT-type an-

gegeben werden.

Beispiele für Satzbestimmungen.

```
-   FILE PLATTE.
-   FILE PLATTE DD   ADD   HINZUFÜGEN
-   FILE PLATTE DEL LÖSCHEN
-   FILE PLATTE DD TYPE XY ÄNDERN

-   FIELD DATUM.
-   FIELD CPGCOM TYP PROG3
```

Feldbestimmungen. 3920

Wird auf den Einsatz von Data Dictionary verzichtet, so müssen zu jeder Satzbestimmung eine oder mehrere Feldbestimmungen codiert werden.

Der Datei-Angabe folgt für jedes auszugebende Feld eine Feldausgabebestimmung. Sie enthält zunächst den Namen des auszugebenden Feldes, soweit es sich um eine Variable handelt. Danach folgt die Ausgabe-Position. Bei Bildschirmdateien wird die Position in der Form 'ZZSS' dargestellt, wobei ZZ die Ausgabezeile und SS die Spalte innerhalb der Zeile angibt.

-	FELD 25.	das letzte Byte der Variablen 'FELD' steht in Position 25
-	X 1240.	Das letzte Byte der Variablen 'X' steht in Zeile 12 Position 40, wenn die Datei eine Bildschirmdatei ist.

Konstanten werden in Hochkommata eingeschlossen und stehen hinter der Positionsangabe.

```
-      542 'TEST'
-     1247 'DAS IST EIN TEXT'
```

Soll das Feld nur unter einer oder mehreren Bedingungen ausgegeben werden, so wird die Anweisung 'ON', gefolgt von bis zu drei Bedingungsschaltern, vorangesetzt.

```
-     ON 15 FELD 25.
-     ON 11 AND NOT 12 AND 13 1242 'TEST'
```

Achtung:
Soll in eine Ausgabe-Position kleiner als 99 eine Konstante unter einer Bedingung ausgegeben werden, zum Beispiel bei 15 in Position 25 die Konstante 'Test', so ist die Kombination

```
-     ON 15 25 'Test'
```

für den Compiler nicht eindeutig, da 25 sowohl eine Bedingung als auch eine Position sein kann. In diesem Falle ist zur Trennung von Bedingungen und Position ein Blank (#) einzutragen.

```
-     ON 15 # 25 'Test'
```

Für Feldbestimmungen ist die folgende Reihenfolge zwingend einzuhalten:

1. Bedingungen eingeleitet durch 'ON' (wahlweise)
2. Feldname (wahlweise)
3. Ausgabe-Position (zwingend)
4. Konstante, Schablone (wahlweise)
5. Schlüsselworte (wahlweise)

Schlüsselworte für Attribute, Editcodes, Farben, EH-Werte, Cursor Usw. können in beliebiger Reihenfolge angehängt werden. Bei den nachfolgend aufgeführten Schlüsselworten genügen jeweils die ersten 3 Buchstaben, z.B. ATT für ATTRIBUTE. Folgende Schlüsselworte gelten :

ATTribut	gefolgt von einem Buchstaben (s. Tabelle, S.7020)
EDItcode	gefolgt von einem Zeichen (siehe Tabelle, S.7070)
Farben:	WHIte, RED, BLUe, GREen, YELlow, PINk, TURqoise
EH Werte	BLInk, REVerse, UNDerscore
Formate	PACked, BINary, HEXadecimal, LOGical
sonstige	CURsor, BLAnk, VARiable-cursor

Das Schlüsselwort 'PAC' gibt an, dass das Feld gepackt ausgegeben wird. 'BIN' steht für binär gepackte Ausgabe und 'HEX' für hexadezimale Ausgabe.

Das Schlüsselwort 'ATT' hinter der Positionsangabe gibt an, dass der folgende Buchstabe ein Feld-Attribut ist. Außerdem können Farbangaben (BLUE, GREEN, PINK, RED, TURQOISE, WHITE, YELLOW) oder Extended Highlight Codes (BLInkend, REVersiv, UNDerlined) folgen.

```
-        FELD 520 ATTRIBUTE A WHITE BLINK.
-        X    630 ATTR N.
```

Das Schlüsselwort 'EDIT' gibt an, dass der folgende Buchstabe ein Aufbereitungsschlüssel ist.

Das Schlüsselwort 'BLAnk' bewirkt, dass das zugehörige Feld nach der Ausgabe gelöscht wird.

Das Schlüsselwort 'CURsor' setzt bei Bildschirmdateien den Positionsanzeiger in die erste Stelle des Feldes. Bei variablem Cursor wird 'VAR' oder 'VARIABLER-CURSOR' als Schlüsselwort eingetragen.

Das Schlüsselwort 'ARRay' ermöglicht es, nicht indizierte Feldgruppen zu beschreiben (indem die Compiler-Fehlermeldung unterdrückt wird).

Die Schlüsselwörter LAST, CONFIRM und NOWAIT für LU6.2 - Verbindungen sind unterstützt. Für Distributed Transaction Processing mit CPG steht eine separate Broschüre zur Verfügung.

Beachte: Es gilt grundsätzlich "Hochkommata vor Schlüsselwörtern"

- z.B. bei hexadezimaler Ausgabe

```
                -     1017 '0000' HEX

          - z.B. bei Konstanten

                -     0211 'KDNR'
                -     0217 '.....' CURSOR  ATTRIBUT N

          - z.B. bei Schablonen

                - NUM 2366 ' . 0 ,  -'  ATTR P  BLUE
```

Beispiele:

```
-   FELD 110.
-   FELD 110 PACKED.
-   2 'TEXT'
-   5 '5C5C00001C' HEXADEZIMAL
-   FELD 110       EDIT2
-   FELD 110 '*'   EDIT2
```

Datenformatierung - Attributbytes für 3270 3950

Ein Feld auf einem Bildschirm vom Typ 3270 ist definiert als der Bereich zwischen zwei Attributbytes.

Vor und hinter jedem Feld und jeder Konstanten, welche auf dem Bildschirm ausgegeben werden, steht ein Attributbyte.

Das ausgegebene Feld bzw. die Konstante endet stets auf der angegebenen Position. Das abschließende Attributbyte des Feldes befindet sich auf der Position hinter der angegebenen Position. Das standardmäßige abschließende Attributbyte hat die Eigenschaft 'geschützt, überspringen'. Durch die Angabe eines nachfolgenden Feldes bzw. einer Konstanten kann dieses Attributbyte durch das Anfangsattribut eines anderen Feldes ersetzt (überschrieben) werden.

Das vor einem Feld stehende Attribut definiert die Eigenschaften des Feldes. Ist in der Ausgabebestimmung mit dem Schlüsselwort 'ATT' kein Attributcode angegeben, so wird standardmäßig das CPG-Attribut 'S' (geschützt, überspringen) angenommen.

Wird in der Ausgabebestimmung weder der Name eines Ausgabefeldes noch eine auszugebende Konstante angegeben, so wird nur das Attribut auf die Bildschirmposition ausgegeben.

Die Möglichkeit, einzelne Attributbytes auf dem Bildschirm auszugeben, kann für folgende Zwecke eingesetzt werden:

a) Felder auf dem Bildschirm definieren, ohne Felddaten ausgeben zu müssen.

b) Das derzeitige Attribut für ein Feld auf dem Bildschirm ändern.

c) Den Cursor auf eine andere Bildschirmposition stellen (hierzu wird ein Attributbyte in Verbindung mit dem Schlüsselwort 'CURsor' angegeben).

Wird das Attribut 'V' für ein Feld oder eine Konstante angegeben, so wird das Anfangsattributbyte für das Feld aus dem einstelligen Feld CPGATR entnommen. Dieses muss im Programm definiert und auf den entsprechenden Hardware-Attributcode für das Feld gesetzt werden, bevor die Ausgabeoperation erfolgt.

Ist 'V' angegeben und enthält die Ausgabebestimmung keinen Feldnamen und keine Konstante, so wird der Inhalt von CPGATR als Attributbyte auf die angegebene Position ausgegeben.

Der CPG-Attributcode 'F' bedeutet, dass das erste Byte des auszugebenden Feldes bzw. der Konstanten als Attributbyte gelten soll. Es muss sich dabei um eine gültige Bitkombination für Attribute für 3270 handeln.

Alphanumerische Daten in der TWA 3952

Alphanumerische Daten können als Konstanten oder als variable Felder auf den Bildschirm ausgegeben werden.

Die Länge der ausgegebenen Daten entspricht der Länge des Feldes in der TWA bzw. der Länge der angegebenen Konstante.

Werden in einer Konstanten '&'-Zeichen bzw. Hochkommata ausgegeben, so sind diese doppelt anzugeben.

Es können Konstanten und Felder ausgegeben werden, die sich über mehrere Zeilen auf dem Bildschirm erstrecken.

Die maximale Länge eines alphanumerischen Feldes ist 256 Bytes.

Numerische Daten 3954

In der TWA gespeicherte Felder werden im Zeichenformat auf den Bildschirm ausgegeben. Vor der Ausgabe werden die Felder entpackt.

Die Länge der Daten auf dem Bildschirm entspricht der TWA-Länge des Feldes (gepackt) multipliziert mit 2 minus 1, d. h. CPG2 reserviert Platz für alle Ziffern, die das Feld enthalten kann.

Wenn ein Feld mit einer geraden Anzahl Ziffern definiert ist, folgt daraus, dass bei der Ausgabe auf dem Bildschirm eine zusätzliche Ziffer vorgesehen werden muss.

CPG unterdrückt nicht die Nullen in numerischen Feldern,die auf diese Weise ausgegeben werden.

Nullenunterdrückung und die Korrektur des abschließenden Alphazeichens sind möglich durch Spezifikation einer Schablone für das Feld oder durch den Eintrag eines Editcodes.

Schablonen 3956

Eine Schablone zur Beschreibung der Ausgabe eines numerischen Feldes kann in Hochkommata direkt hinter der Ausgabeposition angegeben werden Schablonen werden für die folgenden Zwecke eingesetzt:

a) Unterdrückung führender Nullen bis einschließlich zu der angegebenen Position in einem Feld.

b) Die Ausgabe von einem oder mehreren Zeichen rechts von einem Feld, wenn dessen Inhalt negativ ist.

c) Einfügung von Leerzeichen links von dem Feld.

Die Länge der auf den Bildschirm ausgegebenen Daten entspricht der Länge der Schablone (d.h. der Anzahl Zeichen zwischen den Hochkommata) plus 1. Das zusätzliche Zeichen geht den Daten voraus und wird

auf dem Bildschirm als führendes Leerzeichen angezeigt.

Aufbereitete Felder werden von links nach rechts abgearbeitet.Die einzelnen Ziffern des Feldes aus der TWA werden auf die in der Schablone als Leerzeichen oder Null angegebenen Positionen gestellt.Enthält das Feld in der TWA mehr Ziffern, als die Schablone vorsieht,so werden die Daten rechts abgeschnitten.Ungültige Ergebnisse können sich einstellen, wenn die Schablone mehr Ziffern vorsieht als das Feld enthält.

Nullenunterdrückung 3957

Durch Nullen in der Schablone wird die Unterdrückung führender Nullen gesteuert. Alle auftretenden führenden Nullen werden bei der Aufbereitung des Feldes unterdrückt und durch Leerzeichen ersetzt und zwar bis einschließlich zu der Stelle, auf der die Null in der Schablone steht.

Tritt in dem Feld vor der Null eine signifikante Ziffer auf, so endet die Nullenunterdrückung automatisch.

Neben der Unterdrückung führender Nullen unterdrückt diese Funktion auch alle Einfügungszeichen, die als Bestandteil der Schablone vor der Null spezifiziert sind. Wird eine signifikante Ziffer erkannt,so endet die Unterdrückung der Füllzeichen an dieser Stelle.

Ist in der Schablone keine Null angegeben,so gilt Nullenunterdrückung bis zum Ende der Schablone.

Einfügungszeichen 3958

/ Schrägstrich
, Komma
. Punkt sind typische Beispiele für Einfügungszeichen.

Diese werden, wie in der Schablone spezifiziert, ausgegeben, wenn sie nicht wegen führender Nullen in den Eingabedaten unterdrückt werden.

Ein '&' in der Schablone definiert eine Leerstelle (Blank) bei der Ausgabe. Innerhalb einer Schablone muss das '&' nur einmal pro Blank gesetzt werden.

Kennzeichnung negativer Beträge 3959

Die Zeichen, die rechts in der Schablone vor dem abschließenden Hochkomma angegeben sind, werden nur dann ausgegeben, wenn der Inhalt des aufzubereitenden Feldes negativ ist.

Ist der Inhalt des Feldes positiv,werden statt dessen ein oder mehrere Leerzeichen ausgegeben.

Typische Kennzeichnungen für negative Beträge sind '-' und 'CR'.

Einfügung führender Leerzeichen 3960

Alle links in der Schablone angegebenen Einfügungszeichen werden bei der Aufbereitung unterdrückt und durch Leerzeichen ersetzt.

Mit dieser Funktion können bei der Ausgabe auf dem Bildschirm links von einem Feld Leerzeichen eingefügt werden.

Länge der verarbeiteten Daten 3961

Die Anzahl der Ziffern aus dem Feld in der TWA, die in die Schablone übernommen werden,entspricht der erforderlichen Anzahl Bytes für das Feld multipliziert mit 2 minus 1.

Bei Feldern, die mit einer geraden Anzahl Ziffern definiert sind, muss eine zusätzliche linksbündige Zifferposition vorgesehen werden, wenn eine Schablone dafür definiert wird.

Wenn ein numerisches Feld mit Hilfe einer Schablone ausgegeben wird, wird die rechtsbündige Ziffer richtig als Zahl von 0 bis 9 angezeigt.

Beispiele für Schablonen 3963

Ein Feld, das 9-stellig mit zwei Dezimalen definiert ist, soll per Schablone aufbereitet werden.

Schablone	Werte: 0	13,85	- 1234,56
keine	000000000	000001385	000123456
' , '		13,85	1234,56
' 0 , '	0,00	13,85	1234,56
' 0 , -'	0,00	13,85	1234,56-
' 0 , -&!'	0,00	13,85	1234,56- !

Feldgruppen 3965

Aus einer Feldgruppe können sowohl alphanumerische als auch numerische Daten an das Bildschirmgeraet ausgegeben werden.

Soll die gesamte Feldgruppe ausgegeben werden, so muss zur Ausgabe der gesamten Feldgruppe nur deren Name und als Ausgabeposition die letzte Stelle des ersten Elementes angegeben werden.
Die nachfolgenden Elemente werden auf den darunterliegenden Zeilen auf denselben Zeichenpositionen ausgegeben, bis sämtliche Elemente auf dem Bildschirm stehen.

Wird eine Ausgabeposition so angegeben, dass auf dem Bildschirm nicht genügend Zeilen für alle Feldgruppenelemente vorhanden sind, so führt dies zu Fehlern bei der Verarbeitung.

Wird zum Namen der Feldgruppe ein fester Index angegeben, z.B. FG(10), so ist natürlich auch die gezielte Ausgabe einzelner Feldgruppenelemente möglich. In diesem Fall kann und muss jedes Element mit einer eigenen Ausgabebestimmung ausgegeben werden.

Bei numerischen Feldgruppen kann eine Schablone hinter der Position angegeben werden. Entsprechend dieser Schablone wird die Feldgruppe elementweise aufbereitet. Ein Element mit dem Wert Null bewirkt, dass das gesamte Feld bei der Ausgabe nur Leerzeichen enthält.

Der von einem Element belegte Platz ist genauso groß wie bei jedem anderen alphanumerischen oder numerischen Feld mit demselben Modus und derselben Länge.

Überlappte Felder 3967

Die Ausgabebestimmungen für eine Bildschirmdatei werden sequentiell von oben nach unten entsprechend ihrer Reihenfolge im Programm aufgearbeitet.

Daher können Daten so ausgegeben werden, dass sie von einem nachfolgenden Feld bzw. einer Konstante überschrieben werden. Dies führt nicht zu Verarbeitungsfehlern und kann sinnvoll sein, wenn komplexe Bildschirmformate verarbeitet werden.

Feldaufbereitung 3969

Die Feldaufbereitung durch den Aufruf einer EDIT-Operation in der Procedure Division ist in Abschnitt 2420 erklärt.

Die Spezifikationen für die Feldaufbereitung werden als eigene Unterprogramme innerhalb der Output Division behandelt. Sie werden bei der durch einen EXCPT-Befehl eingeleiteten Verarbeitung der Ausgabebestimmungen ignoriert.

Wenn die Verarbeitung einer Ausgabebestimmung für die Feldaufbereitung (mit Hilfe eines EDIT) eingeleitet wird, wird zu den Ausgabebestimmungen für das betreffende Feld verzweigt. Diese werden sequentiell von oben nach unten abgearbeitet, bis entweder die nächste Satzbestimmung im Rahmen der Ausgabebestimmungen oder das Datenende-Kennzeichen für den Quellencode des Programms erreicht wird. An dieser Stelle wird zu der Anweisung in der Procedure Division zurückverzweigt, die unmittelbar auf den EDIT-Befehl folgt.

Schutzsternschreibung 3970

Schutzsternschreibung wird beispielsweise beim Ausstellen von Schecks eingesetzt, um ein nachträgliches Ändern des gedruckten Betrages unmöglich zu machen. Dazu wird das aufbereitete numerische Betragsfeld linksbündig mit Sternen (*) aufgefüllt.

Im CPG2 ist die Schutzsternschreibung zusätzlich zum Aufbereitungsschlüssel unterstützt. Es braucht nur die Konstante '*' anstelle einer Schablone angegeben zu werden.

Beispiel: Das Feld SUMME mit dem Inhalt 98765,55 wird mit Editcode K (Nullenunterdrückung, Komma, Tausenderpunkt, Minuszeichen) und Schutzsternschreibung ausgegeben. SUMME ist als elfstelliges numerisches Feld, davon zwei Dezimalstellen, vereinbart.

Ausgabe: *****98.765,55

Syntax : - SUMME 1020 '*' EDITCODE K

Besonderheit: Der Schutzstern in Kombination mit dem Edit-Code Y für Datumsaufbereitung bewirkt, dass bei leeren Datumsfeldern Blank angezeigt wird und dass führende Nullen unterdrückt werden.

Fließendes Währungszeichen 3972

Eine andere Möglichkeit des Schutzes bietet das fließende Währungszeichen. Dabei wird direkt vor einen Betrag ein Dollarzeichen ($) gesetzt, so dass eine mögliche spätere Manipulation des Betrages verhindert wird.

Das fließende Währungszeichen ist im CPG2 zusätzlich zu den Aufbereitungsschlüsseln unterstützt. Es braucht nur die Konstante '$' anstelle einer Schablone angegeben zu werden.

Beachte: Beim fließenden Währungszeichen wird intern das Ausgabefeld um ein Byte (für das Dollarzeichen) erweitert.

Beispiel: Das Feld SUMME mit dem Inhalt 98765,55 wird mit Editcode K (Nullenunterdrückung, Komma, Tausenderpunkt, Minuszeichen) und fließendem Währungszeichen ausgegeben. SUMME ist als elfstelliges numerisches Feld, davon zwei Dezimalstellen, vereinbart.

Ausgabe: $98.765,55

Syntax : - SUMME 1020 '$' EDI K

Führendes Minuszeichen 3973

In Verbindung mit den Edit-Codes J, K, L und M ist analog zum fließenden Währungszeichen der Eintrag '-' (Minus) unterstützt. Minus sorgt dafür, dass bei der Aufbereitung das Minuszeichen vor die erste Ziffer des Feldes gesetzt wird. Diese Aufbereitung wird z.B. bei der Ausgabe in Microsoft Excel benötigt.

Syntax : - SUMME 10 '-' EDITCODE K

Ergebnis: -3,75

ACCEPT Datensatz direkt lesen 4010

identisch mit CHAIN (s.u.)

ADD Addieren (RPG - Schreibweise) 4015

```
-------------------------------------------------------
(ON) (F1) OP F2 EG (SV) (BD)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   erster Summand
OP   ADD   muss eingetragen werden
F2   zweiter Summand
EG   Ergebnisfeld (Name eines numerischen Feldes)
SV   Service: 'ROUnded', 'RUNden', 'H' zum Runden
BD   bis zu drei Schalter können gesetzt werden
-------------------------------------------------------
```

Beispiele:

```
- A ADD B A                          * A+B=A
- A ADD B A 12                       * Prüfe auf +
- ON 10 A ADD B C                    * Wenn 10: A+B=C
- A       ADD 1 A                    * A+1=A
- A  ADD  -15  A                     * A-15=A
- A ADD 5,67 B RUNDEN                * A+5,67=B,Runden
-   A   ADD  .20      A             * Dezimalpunkt
- FG1    ADD  FG2     FG1           * ADD Feldgruppen
- FG1    ADD  1000    FG1           * Alle + 1000
- FG1    ADD  FG2     FG3           * ADD Feldgruppen
- FG1(1) ADD  FG2(3)  FG1(1)        * ADD Elemente
- FG1(I) ADD  FG2(J)  FG3(K)        * ADD indiziert
- FG1(4) ADD  1000    FG1(4)        * 4.Elem + 1000
-        ADD  B       A             * A+B=A
-        ADD  1000    FG1(4)        * 4.ELEM + 1000
-------------------------------------------------------
```

Zweck:

Zwei Werte sollen addiert werden.

Achtung: Diese Operation kann einfacher in Form einer arithmetischen Formel (C = B + A) geschrieben werden. Vergleiche hierzu die Operation '=' am Ende dieses Abschnitts.

Beschreibung:

Nach Ausführung der Instruktion enthält das Ergebnisfeld (EG) die Summe von Faktor 1 (F1) und Faktor 2 (F2).

F1, F2 und EG können auch den Namen einer Feldgruppe enthalten. In diesem Fall werden alle Elemente dieser Feldgruppe zu den entsprechenden Elementen der anderen Feldgruppe addiert.

Ist die Anzahl der Elemente unterschiedlich, so werden nur so viele Elemente addiert, wie für die kleinste Feldgruppe definiert wurden.

Wird der Name einer Feldgruppe eingetragen, so kann mit einem festen FG(3) FG(7) oder variablen FG(I) FG(KX) Index bestimmt werden, welche Einzelelemente der Feldgruppe zu addieren sind

FG1(1) ADD FELDA FG1(IN).

wobei FELDA der Name eines numerisch definierten Feldes, FG1 der Name einer numerisch definierten Feldgruppe, 1 der feste Index 1 und 'IN' der Name des numerisch definierten Indexfeldes ist, das den variablen Index enthält.

Wenn die Eintragung von F1 und EG gleich ist, so kann die Eintragung in F1 entfallen.

AFOOT Mittelwert einer Feldgruppe rechnen 4020

```
-------------------------------------------------------
(ON) OP F2 EG (SV) (BD)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   AFOOT oder AVERAGE muss eingetragen werden
F2   Name einer numerischen Feldgruppe
EG   Ergebnisfeld (Name eines numerischen Feldes)
SV   Service: 'ROUnded', 'RUNden', 'H'  zum Runden
BD   bis zu drei Schalter können gesetzt werden
-------------------------------------------------------
```

Beispiele:

```
-       AFOOT  FG1  MW                * MW = Mittel FG1
-     AVERAGE  FG1  MW  12            * Ist MW > 0 ?
-------------------------------------------------------
```

Zweck:

Der Mittelwert aller Feldinhalte ungleich Null einer numerischen Feldgruppe soll errechnet werden.

Beschreibung:

Faktor 2 (F2) enthält den Namen der Feldgruppe. Das Ergebnisfeld (EG) enthält den Namen des Feldes, in das der Mittelwert gespeichert werden soll.

Felder mit dem Inhalt 0 werden nicht in die Mittelwertrechnung einbezogen.

```
        AFOOT  FG1  MW

Inhalt der Feldgruppe 'FG1':  FELD 1    125,00
                              FELD 2     75,00
                              FELD 3     85,00
                              FELD 4      0,00
                              FELD 5      0,00
                              FELD 6      0,00

Inhalt des Feldes 'MW'
nach Ausführung der Instruktion:        95,00
```

AVERAGE Mittelwert einer Feldgruppe rechnen 4024

identisch mit AFOOT (s.o.)

BEGAS / ENDAS Assemblercodierte Programmteile 4027

OP BEGAS muss eingetragen werden
OP ENDAS muss eingetragen werden

Beispiel:

```
- BEGAS
      MVC    A,B
- ENDAS
```

Zweck:

Umschalten auf Assembler-Modus bzw. CPG2-Schreibweise.

Beschreibung:

Der Programmierer kann sein Programm an beliebigen Stellen um assembler-programmierte Passagen erweitern. BEGAS schaltet von CPG2 in den Assembler-Modus um; der assembler-codierte Programmteil wird durch die Operation ENDAS beendet und es wird wieder auf CPG2-Logik und CPG2-Schreibweise umgeschaltet.

Für den Assembler-Teil gelten die Regeln der Assembler-Programmierung. Es erfolgt allerdings im CPG-Programm keine Diagnostik. Der Programmierer sollte darauf achten, dass in den Options der Parameter ASS A angegeben wird.

BEGDT Beginn Entscheidungstabelle 4030

```
-----------------------------------------------------
F1 OP
-----------------------------------------------------

F1   Name der Entscheidungstabelle
OP   BEGDT muss eingetragen werden
-----------------------------------------------------

Beispiel:

- DATUM BEGDT.                 * Entsch.-Tabelle DATUM
-----------------------------------------------------

Zweck:

Anfang einer Entscheidungstabelle.

Beschreibung:

Näheres zur Entscheidungstabelle im Kapitel  2410.
```

BEGSR Beginn Unterprogramm 4035

```
-----------------------------------------------------
F1 OP
-----------------------------------------------------

F1   Name des Unterprogramms
OP   BEGSR muss eingetragen werden
-----------------------------------------------------

Beispiele:

- UPRO  BEGSR.                      * Unterprogr. UPRO
-----------------------------------------------------
Zweck:

Anfang eines Unterprogramms.

Beschreibung:

Jedes Unterprogramm muss mit dieser Operation beginnen.
F1 enthält den Namen des Unterprogramms. Alle Unterpro-
gramme müssen  am Ende der  Procedure Division in einer
Subroutine Section liegen, die durch ein  Schlüsselwort
'SUBROUTINES' oder '-S.' eingeleitet werden kann.
```

BITOF Bitschalter löschen 4040

(ON) OP F2 EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP BITOF oder BIT-OF muss eingetragen werden
F2 ein Literal, das die zu löschenden Bits enthält
EG Name eines einstelligen Alphafeldes

Beispiele:

```
-   BITOF '0' BYTE      * Höchstes Bit (Bit 0) aus
-   BITOF '7' BYTE      * niedrigstes Bit (Bit 7) aus
-  BIT-OF '0123' X      * erste 4 Bits aus
-  BIT-OF '0246' Y      * Bits 0 2 4 6 aus
-   ON P1 BITOF '1' PF
```

Zweck:

Löschen einzelner Bits in einem Byte.

Beschreibung:

Mit dieser Operation können 1 bis 8 Bits in einem Byte gelöscht werden. Das Ergebnisfeld enthält den Namen des zu verändernden einstelligen Feldes. Faktor 2 enthält in Hochkommata eingeschlossen die zu verändernden Bits in Form von Ziffern von 0 bis 7 in Zählrichtung von links nach rechts.

Diese Operation gilt nur für alphanumerische Felder.

```
-                  BITOF '567' BYTE
```

Inhalt des Feldes Byte	vorher	nachher
In Bit-Schreibweise:	11110111	11110000
In hexadezimaler Schreibweise:	F 7	F 0
In Characterform:	7	0

BITON Bitschalter setzen 4045

```
-------------------------------------------------------
(ON) OP F2 EG
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   BITON oder BIT-ON muss eingetragen werden
F2   ein Literal, das die zu setzenden Bits enthält
EG   Name eines einstelligen Alphafeldes
-------------------------------------------------------
```

Beispiele:

```
-   BITON '0' BYTE      * Höchstes Bit (Bit 0) ein
-   BITON '7' BYTE      * niedrigstes Bit (Bit 7) ein
-   BITON '0123' X      * erste 4 Bits ein
-   BITON '0246' Y      * Bits 0 2 4 6 ein
-   ON P1 BIT-ON '1' PF
-------------------------------------------------------
```

Zweck:

Setzen einzelner Bits in einem Byte.

Beschreibung:

Mit dieser Operation können 1 bis 8 Bits in einem Byte gesetzt werden. Das Ergebnisfeld enthält den Namen des zu verändernden einstelligen Feldes. Faktor 2 enthält in Hochkommata eingeschlossen die zu verändernden Bits in Form von Ziffern von 0 bis 7 (in Zählrichtung) von links nach rechts.

Diese Operation gilt nur für alphanumerische Felder.

```
-                   BITON '23' BYTE

Inhalt des Feldes BYTE              vorher    nachher
------------------------------      -------- --------
In Bit-Schreibweise:                11000001  11110001
In hexadezimaler Schreibweise:      C   1     F   1
In Characterformat:                 A         1
```

BREAK Eine Schleife beenden 4050

```
-------------------------------------------------------
(ON) OP (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   BREAK muss eingetragen werden
SV   All, um geschachtelte Schleifen zu verlassen
-------------------------------------------------------
```

Zweck:

Eine DO-, DO UNTIL- oder DO WHILE-Schleife soll beendet werden.

Beschreibung am Beispiel:

```
- DO FROM 1 TO WERT WITH I
-    :
-    IF REST <= 0
-       BREAK
-    ENDIF
-    EXSR  SR01
- ENDDO
```

Durch den Befehl BREAK wird eine DO-Schleife unabhängig von der Schleifenbedingung sofort beendet.

BREAK verzweigt hinter das ENDDO. Die Statements zwischen BREAK und ENDDO werden nicht mehr ausgeführt.

BREAK ALL beendet die DO-Verarbeitung in geschachtelten Schleifen, indem hinter das END der äußersten Schleife (bei Subroutinen bis zur äußersten Schleife innerhalb der Subroutine) verzweigt wird.

Beachte:

In einer EVALUATE-Statementgruppe sollte BREAK nicht eingesetzt werden, weil EVALUATE eine eigene DO-Stufe darstellt. Liegt also eine EVALUATE-Gruppe in einer DO-Schleife, so verzweigt das BREAK innerhalb der Gruppe (nur) hinter das END-EVALUATE und nicht hinter das folgende ENDDO.

CAB Vergleichen und verzweigen 4055

```
-----------------------------------------------------
(ON) OP F1 OK F2 (DY) EG (BD)
(ON) F1 OP OK F2 EG (BD)                        (RPG)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CAB  muss eingetragen werden
F1   erstes Vergleichsfeld ( Feldname )
OK   Operator ( s.u. , Tabelle der Operatoren )
F2   zweites Vergleichsfeld ( Feldname oder Konstante )
DY   Dummywort 'TO'
EG   Name eines Anspringpunktes im Programm
BD   bis zu drei Schalter können gesetzt werden
-----------------------------------------------------
```

Beispiele:

```
- CAB A = B ENDE                  * wenn A = B goto ENDE
- CAB A = B ENDE 11 12 13.
- CAB A > B OTTO # 13.
- CAB A GT B OBEN.
- CAB X LESS THAN Y TO OBEN # 50.
- CAB Y >< Y TO NEXT.
- ON P1 AND 15 CAB A GE B TO NEXT # # 22.
```

Zweck:

Zwei Werte sollen miteinander verglichen werden, und auf Grund des Ergebnisses soll zu einem Merkmal verzweigt werden.

Beschreibung:

Die CAB-Operation vergleicht F1 mit F2. Stimmt das Ergebnis der Operation mit dem Operator (OK) der Operation überein, so verzweigt das Programm zu dem in EG gekennzeichneten Label. Anderenfalls wird das Programm mit der nächstfolgenden Operation fortgesetzt.

F1 und F2 müssen einen Feldnamen enthalten, für F2 ist auch ein alphanumerisches oder numerisches Literal gültig. Die Eintragungen in F1 und F2 müssen entweder beide numerisch oder beide alphanumerisch sein.

Sind Ergebnisbezugszahlen spezifiziert, werden sie gesetzt, um das Ergebnis des Vergleichs anzuzeigen, und zwar unabhängig vom verwendeten Operationscode.

Das heisst im einzelnen :

Bezugszahl 1 wird gesetzt, wenn F1 > F2 ist.
Bezugszahl 2 wird gesetzt, wenn F1 < F2 ist.
Bezugszahl 3 wird gesetzt, wenn F1 = F2 ist.

Ist kein Anspringpunkt angegeben, werden die Bezugszahlen gesetzt und die nächste Operation wird verarbeitet.

Tabelle der Operatoren :

Operator		Bedeutung		
GT	>	greater than	F1	größer F2
LT	<	less than	F1	kleiner F2
EQ	=	equal	F1	gleich F2
NE	><	not equal	F1	ungleich F2
GE	>=	greater equal	F1	größer oder gleich F2
LE	<=	less equal	F1	kleiner oder gleich F2

blank Es wird unabhängig vom Vergleich zu dem im Ergebnisfeld angegebenen Label verzweigt. In diesem Fall muss mindestens ein Bedingungsschalter gesetzt sein.

CALL Aufruf von beliebigen Unterroutinen 4060

(ON) OP F2 (EG)

ON Bedingungsabfrage (bis zu drei Schalter)
OP CALL muss eingetragen werden
F2 Phasenname der Unterroutine in Hochkommata
EG Numerisches Feld ohne Dezimalen für Returncode

Beispiele:

- CALL 'PHASE01 '
- CALL 'PHASE07 ' RETURNCODE07
- PARAMETER DATEI
- PARAMETER FUNKTION

Zweck:

Eine beliebige, nicht in CPG geschriebene Unterroutine soll aufgerufen werden. Somit ist die Kommunikation mit allen Softwareprodukten möglich, die eine sogenannte CALL-Schnittstelle besitzen.

Beschreibung:

Das aufzurufende Programm wird in Hochkommata angegeben. Es wird bei der Umwandlung zum CPG-Programm gelinkt.

Optional kann ein numerisches Feld mit null Dezimalstellen angegeben werden. In dieses Feld wird beim Rücksprung aus dem Unterprogramm der Returncode übertragen.

Der Return-Code im CALL-Befehl darf nur dann codiert werden, wenn die Unterroutine tatsächlich einen Return Code setzt; ansonsten können unvorhersehbare Return Codes in das aufrufende Programm gelangen.

Der CALL-Befehl kann erweitert werden um eine beliebige Zahl von PARAMETER- oder PARM-Befehlen zur Parameterübergabe (s.u.).

Beachte: Bei CALL-Aufrufen von COBOL-Programmen setzt man zuerst ein CALL ILBDSET0 ab (damit das COBOL-Programm weiß, dass es ein Unterprogramm ist)

CALLD Zugriff auf Assembler-Datasets (veraltet) 4063

```
-----------------------------------------------------
(ON) F1 OP F2 (SV) BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CALLD muss eingetragen werden
F1   Feldname des Schlüssels
F2   Name eines Datasets
SV   Service: 'U', 'UPDate', 'C', 'CHEck'
BD   ein Schalter (not found) muss gesetzt werden
-----------------------------------------------------
```

Beispiele:

```
- KEY CALLD DATEI 20.                 * 20 = not found
- KDNR CALLD KUNDEN UPDATE 20.
```

Zweck:

Eine Dateizugriffsroutine soll ausgeführt werden; der Service wird an ein Assembler-Dataset übergeben.

Beschreibung:

Mit dieser Operation kann ein benutzerspezifischer Dataset-Zugriff ausgeführt werden. In F1 muss ein Schlüsselfeld eingetragen werden.

Der maximal 8 Stellen lange Dataset-Name muss in F2 eingetragen werden.

Datasets werden heute im Rahmen des CPG3 in CPG programmiert. Die Arbeit mit Assembler-Datasets verliert damit an Bedeutung.

CALLM Unterprogramm aus der Methodenbank abrufen 4065

(ON) OP F2 (EG)

ON Bedingungsabfrage (bis zu drei Schalter)
OP CALLM muss eingetragen werden
F2 Name einer Methodenbank-Routine
EG Parameterliste (Feldname)

Beispiele:

- CALLM UPRO
- CALLM UPRO PARAMETERLISTE
- ON P1 CALLM UPROG PLISTE

Zweck:

Ein Assembler-Programm, das in der Methodenbank katalogisiert ist, soll ausgeführt werden.

Beschreibung:

Mit dieser Operation kann ein benutzerspezifisches Unterprogramm aus der Methodenbank abgerufen werden. F2 enthält den maximal 8-stelligen Namen des Unterprogramms. Das Ergebnisfeld kann den Namen eines Feldes, einer Feldgruppe oder einer Parameterliste enthalten, die im Unterprogramm verwendet werden. Eine Parameterliste kann beispielsweise durch Überlagerung von Feldern in der TWA erzeugt werden.

Beispiel:

```
- PARLST       0 * 14.                  * (1+4+4+5)
-    ME            1.                  * (1 Byte)
-    MENGE         7 0.                * (GEPACKT=4)
-    PREIS         7 2.                * (GEPACKT=4)
-    WERT          9 2.                * (GEPACKT=5)

-    CALLM UPWERT PARLST
```

Im Unterprogramm 'UPWERT' soll der Wert einer Rechnungsposition errechnet werden. Dazu müssen dem Unterprogramm die Felder ME, MENGE und PREIS übergeben werden. Das Ergebnis soll nach Ablauf des Unterprogramms im Feld WERT stehen. Das Feld WERT muss folglich dem Unterprogramm ebenfalls bekannt sein. Die erforderlichen Felder werden dem Unterprogramm in Form einer Parameterliste übergeben.

Voraussetzung ist, dass die Methodenbank eine Assembler-Routine mit dem Namen UPWERT enthält.

CAS(E) Vergleichen und in Subroutine verzweigen 4075

```
-----------------------------------------------------
(ON) OP F1 OK F2 (DY) EG (BD)
(ON) F1 OP OK F2 EG (BD)                        (RPG)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CAS  oder  CASE  muss eingetragen werden
F1   erstes Vergleichsfeld ( Feldname )
OK   Operator ( s.u. , Tabelle der Operatoren )
F2   zweites Vergleichsfeld ( Feldname oder Konstante )
DY   Dummywort 'TO'
EG   Name eines Unterprogramms
BD   bis zu drei Schalter können gesetzt werden
-----------------------------------------------------
```

Beispiele:

```
- CAS  A = B UP01                 * WENN A = B EXSR UP01
- CAS  A = B UP01 11 12 13.
- CASE A > B UP02 # # 13.
- CASE A GT B UPXX.
- CASE X LESS THAN Y TO UPRO # 50.
- CASE Y >< Y TO NEXT.
- ON P1 AND 15 CAS A GE B TO NEXT # # 22.
- CAS  ERROR
- END.                                * obligatorisch
```

Zweck:

Zwei Werte sollen miteinander verglichen werden, und auf Grund des Ergebnisses soll zu einem Unterprogramm verzweigt werden.

Beschreibung:

Die CAS-Operation vergleicht F1 mit F2. Wenn die angegebene Bedingung zwischen den beiden Faktoren besteht, wird das entsprechende Unterprogramm ausgeführt und zum END-Statement der CAS-Gruppe verzweigt. Besteht die Bedingung nicht, so wird die nächste Operation in der CAS-Gruppe ausgeführt.

Eine CAS-Gruppe kann nur CAS-Operationen enthalten und eine END-Anweisung muss die Gruppe abschließen. Nach Ausführung einer Subroutine wird zu der Operation verzweigt, die dieser END-Anweisung folgt.

Die unbedingte CAS-Operation entspricht funktionell der Operation EXSR (s.u.). Alle Operationen, die einer solchen unbedingten CAS-Operation folgen, werden nicht ausgeführt. Somit ist eine solche Operation nur sinnvoll als letzte Anweisung vor dem END einer CAS-Gruppe.

F1 und F2 müssen einen Feldnamen enthalten, für F2 ist auch ein alphanumerisches oder numerisches Literal gül-

tig. Die Eintragungen in F1 und F2 müssen entweder beide numerisch oder beide alphanumerisch sein.

Sind Ergebnisbezugszahlen spezifiziert, werden sie gesetzt, um das Ergebnis des Vergleichs anzuzeigen, und zwar unabhängig vom verwendeten Operationscode.

Das heisst im einzelnen :

Bezugszahl 1 wird gesetzt, wenn F1 > F2 ist.
Bezugszahl 2 wird gesetzt, wenn F1 < F2 ist.
Bezugszahl 3 wird gesetzt, wenn F1 = F2 ist.

Tabelle der Operatoren :

Operator		Bedeutung		
GT	>	greater than	F1	größer F2
LT	<	less than	F1	kleiner F2
EQ	=	equal	F1	gleich F2
NE	><	not equal	F1	ungleich F2
GE	>=	greater equal	F1	größer oder gleich F2
LE	<=	less equal	F1	kleiner oder gleich F2

blank Es wird unabhängig vom Vergleich zu dem im Ergebnisfeld angegebenen Unterprogramm verzweigt.

CBS Vergleichen und in Subroutine verzweigen 4080

Die Operation CBS entspricht weitgehend der Operation CAS sowohl in der Syntax als auch in der Funktionsweise.

Unterschiede zwischen CAS und CBS :

CAS wird in einer Gruppe von CAS-Statements verarbeitet und verzweigt nach dem eventuellen Unterprogrammsprung zum END-Statement der CAS-Gruppe. Es wird also maximal ein Unterprogramm aus einer CAS - Gruppe angesprungen.

CBS dagegen wird nicht in Gruppen verarbeitet. Ein zum CBS gehörendes END-Statement ist nicht unterstützt Es wird immer nach Abarbeitung des CBS-Befehls zum nächstfolgenden Statement verzweigt. Besteht eine Folge von n Statements aus n CBS-Befehlen , dann sind (im Gegensatz zu CAS) auch n verschiedene Unterprogrammsprünge möglich.

CHAIN Satz wahlfrei lesen 4085

```
-------------------------------------------------------
(ON) F1 OP F2 (KW EG) (SV) (SV) (BD)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CHAIN, ACCEPT oder GET-UPDATE ist einzutragen
F1   Feldname des Schlüssels
F2   Dateiname
KW   Schlüsselwort  BRANCH  zum Verzweigen
EG   Anspringpunkt
SV   Service: 'U', 'UPDate', 'C', 'CHEck','P'
BD   ein Schalter (not found) kann gesetzt werden
-------------------------------------------------------
```

Beispiele:

```
- KEY CHAIN DATEI
- KDNR ACCEPT KUNDEN UPDATE CHECK 20.  * 20 = not found
- KEY  GET-UPDATE DATEI3   BRANCH NOTFOUND  **.
-------------------------------------------------------
```

Zweck:

Ein Satz einer Datei soll wahlfrei gelesen werden.

Beschreibung:

Mit dieser Operation wird ein Satz der in F2 genannten Random-Datei mit dem in F1 genannten Schlüssel gelesen.

Die Länge des Schlüssels muss mit der in der FILES-Division angegebenen übereinstimmen.

Der Schalter in BD wird vom Programm gesetzt, wenn der gesuchte Satz nicht gefunden wurde. Wurde kein Schalter angegeben, wird die Information 'NF' für not found im internen Feld CPGFRC übergeben.

In EG kann ein Label in der Form BRANCH LABEL angegeben werden, zu dem verzweigt wird, falls der Satz nicht gefunden wurde; soll nur verzweigt und kein Schalter gesetzt werden, kann statt eines Schalters '**' in BD eingetragen werden.

Ein 'C' oder 'CHEck' in SV bedeutet, dass der Satz nur auf Verfügbarkeit geprüft wird. In diesem Falle werden keine Daten gelesen. Der Schalter in SV wird auf 'Ein' gesetzt, wenn der Satz nicht gefunden wurde.

'UPDate' oder 'U' in SV bewirkt, dass der Satz solange für ein weiteres CHAIN für Update gesperrt wird, bis ein Update durchgeführt oder der Satz durch ein RNDOM wieder freigegeben wird. Es wird lediglich auf Satz-Ebene gesperrt. Bei Share-Option 4 oder Journaling wird das gesamte VSAM-CI gesperrt.

Ein 'P' in SV schließt CHECK und UPDATE ein. Die explizite Angabe beider Services ist ebenfalls unterstützt.

Beachte:

Befindet sich die Datei im sequentiellen Zugriffsmodus, dann arbeitet CHAIN wie die Operation SETLL. Es wird nur beim angegebenen Schlüssel in der Datei positioniert, Daten werden nicht eingelesen.

Bei Dataset-Verarbeitung benötigt die CHAIN-Operation immer Input-Bestimmungen. Bei VSAM-Dateiverarbeitung ist CHAIN auch ohne Input-Bestimmungen unterstützt.

CHANG(E) Feldinhalte austauschen 4090

```
-------------------------------------------------------
(ON) F1 OP F2
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Feldname
OP   CHANG oder CHANGE muss eingetragen werden
F2   Feldname
-------------------------------------------------------
```

Beispiele:

- A CHANG B.
- OTTO CHANGE HUGO.

Zweck:

Die Inhalte zweier Alphafelder sollen ausgetauscht werden.

Beschreibung:

Mit dieser Operation können die Inhalte zweier Datenfelder ausgetauscht werden.

```
-           FELD1 CHANG FELD2
```

Nach Ausführung der Operation enthält Feld1 den Wert von Feld2 und Feld2 den Wert von Feld1.

```
Sonderfall: - FELD CHANGE FELD. * Nach der Ausführung
            -                   * ist FELD gleich X'00'.
```

CHECK Dateistatus prüfen 4095

```
-----------------------------------------------------
(ON) OP F2 (SV) (BD)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CHECK  muss eingetragen werden
F2   Dateiname
SV   Service: 'VAR', variabler Dateiname
BD   Schalter können gesetzt werden
-----------------------------------------------------

Beispiele:

- CHECK DATEI 20.                    * 20 = open
- CHECK FILE VAR 20 21.              * 21 = not in FCT
-----------------------------------------------------

Zweck:

Der Status einer Datei (open, closed) soll geprüft wer-
den, bzw. ob die Datei in der FCT definiert ist.

Beschreibung:

Im F2 wird der Name der zu prüfenden Datei eingetragen.
In BD kann ein Schalter eingetragen sein. Dieser Schal-
ter wird gesetzt, wenn die Datei vom System her eröff-
net ist. Ist der Schalter nicht gesetzt, so ist die Da-
tei geclosed.

Zusätzlich dazu kann mit einer zweiten Bezugszahl ge-
prüft werden, ob die Datei in der CICS FCT eingetragen
ist oder nicht. Der  zweite Schalter wird gesetzt, wenn
die Datei nicht in der FCT definiert wurde.

Will man auf Schalter verzichten, so kann man die Infor-
mationen 'NO' (not open) und 'NF' (not found) im  inter-
nen Feld CPGFRC abfragen. Für den Zustand OPEN DISABLED
kann unterschieden werden, ob dafür OD oder Blank in
CPGFRC zurückgemeldet wird (siehe CPGURSI2 im Handbuch
CPG3-Installation). Der Default ist ab Release 2.5  OD.

In Batchanwendungen wird 'NF' gesetzt, wenn im Hauptpro-
gramm die Datei nicht als File definiert ist und wenn
bei der Ausführung diese Datei noch nicht eröffnet
wurde.

In SV kann ein  "V"  eingetragen werden.  In diesem Fall
wird in F2 der Name eines achtstelligen Feldes eingetra-
gen,  in dem zur  Ausführungszeit  der Dateiname  abge-
stellt wird.

Beispiel:

-               CHECK CPGWRK 50

Ist nach Durchführen der Operation Schalter 50 gesetzt,
so ist diese Datei zur Zeit eröffnet.
```

CLEAR Feldinhalte löschen 4100

(ON) OP

ON Bedingungsabfrage (bis zu drei Schalter)
OP CLEAR muss eingetragen werden

Beispiel:

- CLEAR

Zweck:

Alle Felder des Programms sollen gelöscht werden.

Beschreibung:

Diese Operation löscht alle alphanumerischen Felder auf Blank und alle numerischen Felder auf Null.

Achtung: Wir empfehlen, vorher alle Dateien mit dem Befehl - RNDOM *ALL freizugeben.

Einschränkung : CLEAR ist nicht unterstützt
- in HL1-Modulen
- in Batch-Programmen
- in Subroutines

CLEAR-IND Schalter abhängig von einem Index löschen 4105

identisch mit CLRIN (s.u.)

CLOSE Datei abschließen 4110

```
-----------------------------------------------------------
(ON) OP F2 (SV) (BD)
-----------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CLOSE  muss eingetragen werden
F2   Dateiname
SV   Service: 'V', variabler Dateiname
BD   ein Schalter für 'Not open'
-----------------------------------------------------------

Beispiele:

- CLOSE DATEI  20.                * 20 = not open
- CLOSE FILE V 20.
-----------------------------------------------------------

Zweck:

Eine Datei soll abgeschlossen werden.

Beschreibung:

Die explizite CLOSE-Operation schließt eine Datei ab.
F2 bezeichnet die abzuschließende Datei.

In BD kann ein  Schalter angegeben werden, der  nach er-
folgreichem  Abschluss der CLOSE-Operation auf 'Ein' ge-
setzt wird.

Soll auf den Schalter verzichtet werden, so  kann im in-
ternen Feld CPGFRC 'NC' für 'not closed'  und 'NF' für
'nicht in der FCT gefunden' abgefragt werden.

Ein 'V' in SV bedeutet, dass in F2 ein Feldname eingetra-
gen ist. Das Feld muss acht Stellen alpha definiert sein
und einen gültigen Dateinamen enthalten.

-             MOVEL 'DATEI' TO FELD
-             CLOSE FELD VARIABEL
```

CLRIN Schalter abhängig von einem Index löschen 4115

(ON) OP EG (SV) BD

ON Bedingungsabfrage (bis zu drei Schalter)
OP CLRIN oder CLEAR-IND muss eingetragen werden
EG Numerisches Indexfeld mit 0 Dezimalstellen
SV 'INDex', um die Indexprüfung zu unterdrücken
BD ein Schalter

Beispiel :

- CLEAR-IND INDEX 25

Zweck:

Löschen eines Schalters in Abhängigkeit von einem Wert

Beschreibung:

Der angegebene Schalter ist der erste in einer fortlaufenden Bezugszahlengruppe.

Nach Ausführung der Operation wird die Bezugszahl gelöscht, die sich aus der Addition der angegebenen Bezugszahl und dem Inhalt des Indexfeldes verringert um 1 ergibt. (Wäre also im obigen Beispiel der Wert von INDEX gleich 4, dann wäre nach dem CLRIN der Schalter 28 gelöscht.)

Enthält EG keinen gültigen Index, so wird bei der Ausführung die Fehlermeldung 'INDEX FALSCH' ausgegeben.

Die Service-Funktion 'INDex' unterdrückt die Indexprüfung. Wenn das Indexfeld gleich Null ist, wird die Fehlermeldung 'INDEX FALSCH' nicht angezeigt und die Verarbeitung fortgesetzt.

COMP(ARE) Vergleichen (in alter RPG-Logik) 4120

```
-----------------------------------------------------
(ON) F1 OP F2 (KW EG) BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   COMP oder COMPARE muss eingetragen werden
F1   Feldname
F2   Feldname
KW   Schlüsselwort 'BRANCH', 'GO', 'GOTO'
EG   Anspringpunkt
BD   mindestens ein Schalter muss gesetzt werden
-----------------------------------------------------
```

Beispiele:

```
- A COMP B 11.                          * 11 = A > B
- A COMPARE B # 12 13.
- NR COMP 4711 BRANCH FEHLER 11 11.
- UST COMPARE  0,15  BRANCH A1987  << ==
- ERRORCODE  COMP  'NF'  BRANCH  FEHLER  EQ
```

Zweck:

Die Inhalte zweier Datenfelder sollen miteinander verglichen werden. Auf Grund des Ergebnisses sollen Schalter gesetzt und - falls gewünscht - soll das Programm an anderer Stelle fortgesetzt werden.

Beschreibung:

F1 und F2 werden miteinander verglichen. Wenn F1 größer, kleiner oder gleich F2 ist, wird der entsprechende Schalter in BD angesetzt. Die Position bestimmt den Wert des Schalters in der oben genannten Reihenfolge. Wird nur ein Schalter angegeben, so hat dieser die Bedeutung 'größer'. Soll ein Schalter bei 'gleich' gesetzt werden, so müssen die anderen Positionen durch Blanks (#) markiert werden.

```
        A  COMP  B  # # 11.
```

Vor Ausführung der Operation werden diese Schalter gelöscht.

Die zu vergleichenden Felder müssen entweder beide numerisch oder beide alphanumerisch definiert sein.

In EG kann nach einem Schlüsselwort 'BRANCH' oder 'GO' oder 'GOTO' ein Merkmal eingetragen werden. Das Programm verzweigt dann zu diesem Merkmal und setzt dort die Verarbeitung fort, wenn eine der Bedingungen erfüllt ist.

Dabei können für diese Operation ausnahmsweise auch an Stelle einer Bezugszahl zwei Sterne eingetragen werden. Enthält z.B. die dritte Position von BD zwei Sterne, so wird unmittelbar zu dem im Ergebnisfeld eingetragenen Merkmal verzweigt, wenn F1 und F2 gleich sind.

Zur besseren Übersicht können statt der Sterne auch die Zeichen '>>', '<<', '==' oder 'GT', 'LT', 'EQ' eingetragen werden.

Im CPG2-Format gibt es die Schlüsselwörter 'DAT' bzw. 'DATI' für COMP, also z.B. A COMP B DATI 10 20 30.

COMRG Communication Region 4125

COM-REG Communication Region 4126

```
-------------------------------------------------------
(ON) OP EG
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   COMRG  oder COM-REG muss eingetragen werden
EG   Name eines 32-stelligen Alphafeldes oder CPGSIN
-------------------------------------------------------
```

Beispiele:

- COMRG FELD.
- COM-REG CPGSIN

```
-------------------------------------------------------
```

Zweck:

Systeminterne Daten sollen dem Programm zur Verfügung gestellt werden.

Beschreibung:

1. COMRG mit beliebigem 32-stelligen Alphafeld

Die Operation COMRG stellt eine Communication Region zur Verfügung, die dem Programmierer den Zugriff auf interne Daten des TP-Monitors ermöglicht. Die Communication Region ist ein 32-stelliges alphanumerisches Feld, das wie folgt belegt ist:

```
Stelle  1 -  3   Operator Identification (Sign On Table)
        4 -  6   Cursor Position numerisch
                 gepackt (170=Zeile 3 Position 11)
        7 - 10   Transid
       11 - 12   Anzahl Bildschirmz. Default   (gepackt)
       13 - 14   Anzahl Bildschirmsp.Default   (gepackt)
```

```
15 - 16   Anzahl Bildschirmz. Alternate (gepackt)
17 - 18   Anzahl Bildschirmsp.Alternate (gepackt)
19 - 21   Tasknummer (gepackt)
22 - 22   'U' = UCTRAN     Übersetzung in Groß-
          'N' = NOUCTRAN   buchstaben ( ja/nein )
23 - 26   CPG intern
27 - 27   UCTRAN Informationen.
          'N' = NOUCTRAN, keine Übersetzung in
                Großbuchstaben
          'T' = UCTRAN Transaktion (ab CICS 2.2)
          'U' = UCTRAN, Übersetzung in Großbuch-
                staben
28 - 32   Reserve
```

Das Ergebnisfeld muss den Namen eines 32-stelligen Feldes enthalten, das die Communication Area aufnimmt. Die einzelnen Felder können über eine 'SELCT' Operation ausgewählt werden, bzw. durch Unterdefinition in der Data Division.

Beispiel:

```
- -D. FELD 0 * 32.             *  GESAMT
-             OPID   3.       *  OPERATOR ID
-             CURSOR 5 0.     *  CURSOR POSITION
-             TRANID 4.       *  TRANSID
-             DZEILE 3 0.     *  ANZ. BILDSCHZ. DEF.
-             DSPALT 3 0.     *  ANZ. BILDSCHSP.DEF.
-             AZEILE 3 0.     *  ANZ. BILDSCHZ. ALTERN.
-             ASPALT 3 0.     *  ANZ. BILDSCHSP.ALTERN.
-             TSKNO  5 0.     *  TASK NUMMER
-             UCTRN  1.       *  UCTRAN BYTE
-             FILLER 4.       *  CPG intern
-             UCTRNE 1.       *  UCTRAN BYTE erweitert
-             REST   5.       *  noch nicht belegt

- -C.         COMRG FELD
```

2. COMRG mit dem internen Feld CPGSIN

Mit COMRG CPGSIN können erweiterte Systeminformationen abgerufen werden.

Ein grundsätzlicher Unterschied besteht in der Verarbeitungslogik: CPGSIN wird in der Input Division als Feld wie für eine SELECT-Eingabe beschrieben. COMRG füllt automatisch die in der Input Division beschriebenen Felder.

Folgende Parameter können abgerufen werden:

```
- -I. FIELD CPGSIN.     * Der   Bereich  wird  nach  dem
-                       * COMRG wieder  freigegeben !
-       1   8   APPLID. * CICS Application Id
-       9  12   SYSID.  * CICS SYSID
-      13  20   NETNAM. * VTAM Net Name
-      21  28   USERID. * achtstellige User Id
-      29  31   OID.    * dreistellige Operator Id
-      32  34   OPCLAS. * Hex: OPCLASS, Security-Klasse
-      35  37   OPSEC.  * Hex: OPSECURITY, alter Securi-
-                       *                    ty Key
-      38  45   OPKEYS. * Hex: OPERKEYS, neuer Security-
-                       *                 Key
- PAC  46  48 0 CWA.    * Länge der Common Work Area
- PAC  49  51 0 TWA.    * Länge der Transaction Work A.
- PAC  52  54 0 TCTUAL. * Länge der Terminal User Area
-      55  58   TERMNL. * Terminal-Features COLOR,EXTDS,
-                       *                 HILIGHT und PS
- PAC  59  61 0 ACTROW. * akt. Anzahl Bildschirmzeilen
- PAC  62  64 0 ACTCOL. * akt. Anzahl Bildschirmspalten
-      65  68   ABCODE. * letzter ABEND Code
-      69  70   STCODE. * QD = Start von Transient Data
-                       * S  = Start mit EXITI
-                       * SD = Start mit EXITD
-                       * TD = Start über Bildschirm-
-                       *      eingabe
-      71  80   DAY.    * Wochentag im Klartext
- PAC  81  81 0 DAYNO.  * Wochentag als Ziffer ( Sonntag
-                       * = 0, Samstag = 6 )
- PAC  82  84 0 COMLEN. * Länge der Common Area
-      85  86   PFKEY.  * Programmfunktionstaste
- PAC  87  90 0 TASKNO. * CICS Tasknummer
-      91  94   TRANID. * Transaction Identification
- PAC  95 102 0 TIME1.  * Anzahl Sekunden seit 1.1.1900
-     103 108   D1.     * Datum JJ.TTT
-     109 116   D2.     * Datum JJ.TT.MM
-     117 124   D3.     * Datum JJ.MM.TT
-     125 132   D4.     * Datum TT.MM.JJ
-     133 140   D5.     * Datum MM.TT.YY
- PAC 141 148 0 TIME2.  * Anzahl Tage seit 1.1.1900
- PAC 149 151 0 YEAR.   * aktuelles Jahr
- PAC 152 153 0 CURROW. * Zeile der Cursorposition
- PAC 154 155 0 CURCOL. * Spalte der Cursorposition
```

CONTINUE Unterbrechen einer Schleife 4130

```
-----------------------------------------------------------
(ON) OP
-----------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CONTINUE oder CONT  muss eingetragen werden
-----------------------------------------------------------
```

Zweck:

Eine DO-, DO UNTIL- oder DO WHILE-Schleife soll unterbrochen, aber entsprechend der Schleifenbedingung weiter durchlaufen werden.

Beschreibung am Beispiel:

```
- DO FROM 1 TO WERT
-    :
-    IF FELD = 0
-       CONTINUE
-    ENDIF
-    EXSR  SR02
- ENDDO
```

Die Schleife wird entsprechend der Schleifenbedingung durchlaufen. Im Fall FELD = 0 wird die Subroutine SR02 jedoch nicht ausgeführt.

CONTINUE verzweigt zurück zum zugehörigen DO-Befehl.

CONVERT Konvertieren eines alphanumerischen Feldes 4135

CONVT Konvertieren eines alphanumerischen Feldes 4136

```
-----------------------------------------------------
(ON) OP (F2 INTO) EG (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   CONVERT oder CONVT  muss eingetragen werden
F2   Name des zu konvertierenden Feldes
EG   Ergebnisfeld ( INTO muss codiert werden )
SV   CHAracter,DATe,HEX,LOW,SEConds,TIMe,Year,UDAte
-----------------------------------------------------
```

Beispiele:

- CONVERT A1
- CONVERT A1 INTO A2 HEXADECIMAL

Zweck:

Der Inhalt eines alphanumerischen Feldes soll konvertiert werden. Bei der Konvertierung von Zeit- und Datum-Informationen sind auch numerische Operanden erlaubt.

Beschreibung:

CONVERT A1 übersetzt in A1 Kleinbuchstaben in Großbuchstaben.

Die Servicefunktionen haben folgende Auswirkung:

LOW (Lower Case Translation) übersetzt Großbuchstaben in Kleinbuchstaben.

HEX übersetzt Character in ihre Halbbytes (EBCDIC-Darstellung), z. B. den Buchstaben A in C1.

CHA fasst je zwei Hexadezimalwerte zu einem Character zusammen, z. B. C1 zu A.

Datumskonvertierung (DATe und Year):

Bei Date und Year müssen die beiden Operanden von gleicher Länge sein. Folgende Längen sind unterstützt:

alpha	6	num. 6,0 oder 7,0	Inhalt:	(0)TTMMJJ
alpha	8	num. 8,0 oder 9,0	Inhalt:	(0)TTMMJJJJ
alpha	8		Inhalt:	TT.MM.JJ
alpha	10		Inhalt:	TT.MM.JJJJ

DAT vertauscht Jahr und Tag, so dass die Jahreszahl nach der Operation vorne im Feld steht.

Y für Year wird benötigt, falls die Jahreszahl in den ersten vier Stellen des Feldes steht. YEAR ist die Umkehrfunktion von DATE. (Bei zweistelligen Jahreszahlen genügt die Servicefunktion DATe für beide Konvertierungsrichtungen).

Datumskonvertierung vom ISO-Format ins UDATE-Format.

UDA vertauscht Jahr und Tag, so dass die Jahreszahl nach der Operation hinten im Feld steht und bereitet das Feld mit '.' auf.

```
-      CONVERT  F2  into  EG  UDAte

F2:
  numerisch        8,0 und 9,0  Inhalt:  (0)JJJJMMTT
  alphanumerisch   8            Inhalt:     JJJJMMTT
EG:
  alphanumerisch   8            Inhalt:     TT.MM.JJ
  alphanumerisch  10            Inhalt:   TT.MM.JJJJ
```

Konvertierung: Uhrzeit <-> Sekunden (SECs, TIME)

Die Uhrzeit steht in einem Feld, das numerisch von 5,0 bis maximal 9,0 definiert ist. Die beiden letzten Stellen werden als Sekunden interpretiert, die vorletzten als Minuten und alle weiteren als Stunden. Die maximale Anzahl Sekunden, die bei der Konvertierung unterstützt ist, ist 359.999.999 in einem mit 9,0 definierten numerischen Feld.

SECs übersetzt eine Uhrzeit bzw. einen Wert (in Stunden, Minuten und Sekunden) in die Sekunden-Anzahl. Ist die Sekunden- oder Minutenanzahl größer 59, so wird das Ergebnisfeld auf Null gesetzt.

TIME übersetzt eine Sekunden-Anzahl in einen Zeit-Wert.

Das Ergebnisfeld ist optional. Es muss jedoch bei den Services HEX, CHA, SECS und TIME vorhanden sein. Zur Kennzeichnung des Ergebnisfeldes muss das Schlüsselwort INTO angegeben werden.

Bei der Verarbeitung mit EG bleibt das Ursprungsfeld in F2 unverändert. Die Konvertierung wird linksbündig in der Länge des kürzeren Operanden durchgeführt.

Für die Eintragungen in F2 und EG gilt:
- Konstanten sind nicht erlaubt
- die Operation ist in beiden Einträgen indizierbar
- wenn beide Einträge vorgenommen werden und einer der beiden eine nicht indizierte Feldgruppe ist, muss auch der andere eine nicht indizierte Feldgruppe sein.

COPY Assembler Unterprogramm einfügen 4140

OP F2

OP 'COPY' muss eingetragen werden
F2 Name des Assembler-Copybooks

Beispiel:

- COPY ASSBOOK.

Zweck:

Ein Assembler Copybook soll ins Programm eingefügt werden.

Beschreibung:

Die Instruktion entspricht der COPY-Instruktion des Assembler. Sie erlaubt dem Programmierer, katalogisierte Assembler-Quellenmodule in das Programm einzufügen. F2 enthält den Namen des Quellenmoduls und darf maximal 8 Stellen lang sein.

Beispiel: COPY ASSBOOK

Das in der Source-Library katalogisierte Assembler-Book wird an dieser Stelle in das CPG-Programm eingefügt.

DEBUG Testhilfe-Funktion 4150

```
-----------------------------------------------------
(ON) OP (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   'DEBUG' muss eingetragen werden
SV   OFF und ON, um QDF-Tests zu unterbrechen
-----------------------------------------------------
```

Beispiele:

- DEBUG
- DEBUG OFF

Zweck:

1. Ein Test-Bild (bei Macro Level) oder ein Screen Dump bei Command Level) wird am Bildschirm angezeigt.
2. Die Möglichkeit, mit QDF zu testen, wird für einen Programmabschnitt aufgehoben bzw. reaktiviert.

Beachte:

Zu 1: Die DEBUG-Funktion wird auch von der interaktiven Testhilfe QDF (Quick Debugging Facility) erfüllt. DEBUG verliert somit als CPG-Operation an Bedeutung.

Zu 2: Wird DEBUG ON/OFF codiert, so können Programmteile vom interaktiven Test mit QDF ausgeschlossen werden. Diese Funktion ist zum Beispiel dann vorteilhaft, wenn große Programme getestet werden sollen, die infolge des Option-Parameters DEBug so groß werden, dass die Assemblierung Addressability Errors bringt. Für den verriegelten Programmteil wird kein Mehr-Code generiert.

DELC Zeichen entfernen 4155

```
-----------------------------------------------------
(ON) OP F2 EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   'DELC' muss eingetragen werden
F2   das zu entfernende Zeichen als Literal
EG   Name eines Alphafeldes oder -feldgruppenelements
     oder Name einer alphanumerischen Feldgruppe
-----------------------------------------------------
```

Beispiele:

```
-   DELC '0'  FELD
-   ON PF10 DELC X'FF' INPUT
-----------------------------------------------------
```

Zweck:

Löschen eines Zeichens aus einem alphanumerischen Feld.

Beschreibung:

Mit dieser Operation kann ein Zeichen aus einem alphanumerischen Feld gelöscht werden. Dabei werden alle folgenden Zeichen um eine Stelle nach links verschoben und in der letzten Stelle wird ein Blank eingeschoben. Bei einstelligen Feldern ist DELC nicht unterstützt.

Hinter dem Operationscode DELC steht das zu entfernende Zeichen in Hochkommata oder hexadezimal in der Form X'00', dahinter der Namedes zu bearbeitenden Feldes.

Beispiel : - DELC '*' FELD

```
Feldinhalt vorher  :        'A*B**C'
Feldinhalt nachher :        'ABC   '
```

Es kann auch eine alphanumerische Feldgruppe verarbeitet werden. DELC arbeitet in diesem Fall element-übergreifend.

Beispiel : - DELC '-' FG36

```
Feldinhalt vorher  :        'A-B-C-'
                            '------'
                            'DEFGHI'

Feldinhalt nachher :        'ABCDEF'
                            'GHI   '
                            '      '
```

DELET(E) Satz von einer Plattendatei löschen 4160

```
(ON) (F1) OP F2
```

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Schlüsselfeld, das den zu löschenden Satz angibt
OP DELET oder DELETE muss eingetragen werden
F2 Name der Datei, auf der gelöscht wird

Beispiel :

```
-   KEY  DELETE  CPGWRK
```

Zweck:

Löschen eines Satzes in einem Datenbestand, ohne in die Output Division zu verzweigen.

Beschreibung:

Mit DELET wird ein Satz in einem Datenbestand gelöscht, so dass er nicht wieder aufgefunden werden kann.

Vor dem DELET kann ein Schlüsselfeld angegeben werden, dessen Inhalt den zu löschenden Satz bestimmt. Wird ein solches Feld nicht angegeben, so löscht DELET den Satz, der zur Zeit in Bearbeitung ist. Hinter dem Operationscode steht der (bis zu achtstellige) Name der Datei, von der der Satz gelöscht werden soll.

DEQ(UEUE) Programmteil entriegeln 4165

```
------------------------------------------------------
F1 OP (SV)
------------------------------------------------------

F1   Merkmal, gleicher Name wie beim Verriegeln ( ENQ )
OP   DEQ oder DEQUEUE muss eingetragen werden
SV   Mögliche Eintragung :  EXTern
------------------------------------------------------

Beispiele:

-   A100   DEQUEUE
-   X  DEQ  EXTERN
- * siehe auch : ENQ
------------------------------------------------------
```

Zweck:

Entriegeln eines gesperrten Programmteils.

Beschreibung:

Mit DEQ wird ein mit ENQ gesperrter Programmteil wieder entriegelt. Zum Entriegeln gesperrter externer Speicheradressen wird die Servicefunktion 'EXTern' angeboten.

Eine ausführliche Beschreibung des Verriegelns wird bei ENQ mit Beispiel gegeben.

DISPLAY Konsolmeldung ausgeben 4170

identisch mit DSPLY

DIV Dividieren (RPG - Schreibweise) 4175

Zweck:

Zwei Werte sollen dividiert werden.

Achtung: Diese Operation kann einfacher in Form einer arithmetischen Formel (C = B / A) geschrieben werden. Vergleiche hierzu die Operation '=' am Ende dieses Abschnitts.

Die Operation DIV wird syntaktisch genauso wie ADD behandelt. Entnehmen Sie Grammatik und Beispiele bitte sinngemäß der Beschreibung der Operation ADD.

DLI DL/I-AUFRUF 4180

```
------------------------------------------------------
(ON) F1 OP F2 EG LG (SV) BD
------------------------------------------------------

ON   Bedingungsabfrage ( bis zu drei Schalter )
F1   gültiger DL/I-Call
OP   DLI oder DL1 muss eingetragen werden
F2   PSB-Name in Hochkommata oder '#' ( Nummernzeichen )
EG   Name des Ein-/Ausgabebereichs
LG   Nummer des verwendeten PCB
SV   V  für variable Einträge
BD   eine Bezugszahl als Fehlerschalter
------------------------------------------------------
```

Beispiele:

```
-   GHU DLI # ARTIRT 3 10
-   GU DL1 # MYROOT 2 VARIABEL  10
```

Zweck:

Call zur DL/I Datenbank

Beschreibung:

F1 enthält einen gültigen DL/I-Call.

F2 bleibt in der Regel frei, das heisst es muss ein '#' (Doppelgitter) eingegeben werden. F2 kann aber auch in Hochkommata einen PSB-Namen enthalten, wenn in einem Programm auf verschiedene PCBs zugegriffen wird.

In EG wird die Datenbeschreibung der DL/I eingetragen, z.B. bei der Eingabe der Name einer in der Input Division beschriebenen Datenstruktur.

In LG wird die Nummer des verwendeten PCBs angegeben.

In BD muss ein Schalter eingetragen werden. Dieser wird gesetzt, wenn der Call nicht ordnungsgemäß verlaufen ist, d.h. wenn der Returncode von DL/I nicht gleich ' ', 'GA' oder 'GK' gesetzt wird. Der genaue DL/I-Returncode kann über das CPG-interne Feld CPGDRC abgefragt werden.

Durch die Operation DLI werden alle aufgebauten SSAs gelöscht. Für jeden weiteren DL/I Aufruf sind die SSAs mit den Operationen QSSA oder USSA neu zu setzen.

Der Service V bewirkt, dass die Werte der einzelnen Einträge dem Feld CPGDLV entnommen werden.

DO Befehlsfolge ausführen 4185

```
-----------------------------------------------------
(ON) OP (OE) (DY) (F1) (OK) (DY) (F2) (DY) (EG) (SV)
oder
(ON) OP OE F1 OK F2 (BO)
-----------------------------------------------------
```

ON	Bedingungsabfrage (bis zu drei Schalter)
OP	DO muss eingetragen werden
OE	UNTIL, WHILE, FROM können eingetragen werden
F1	erstes Vergleichsfeld
OK	Operator (s.u., Tabelle der Operatoren)
F2	zweites Vergleichsfeld
EG	Index. Name eines numerischen Feldes mit 0 Dez.St.
SV	Service: 'LOOP'
DY	Dummywort ('TO' 'TIMES' 'WITH')
BO	Boolesche Operation AND oder OR

```
-----------------------------------------------------
```

Beispiele:

```
- DO.                    * Befehlsfolge ausführen
- DO 10.                 * 10 mal ausführen
- DO 10 TIMES.                      * 10 mal
- DO 5 TIMES WITH I.                * mit Index I
- DO FROM X TO Y WITH I.            * Anfangswert X
- DO LOOP.                          * Endlos-Schleife
- DO LOOP WITH I
- DO WHILE A = B.                   * solange A = B
- DO UNTIL X = 0.                   * bis X = 0
- DO WHILE A EQ FGA(I)
- DO WHILE A EQUAL B.
- DO WHILE FGN(3) IS EQUAL TO ZAHL.
- DO WHILE A IS NOT GREATER THAN B.
- DO WHILE A > B AND X = 0
- DO UNTIL A = B OR A GE 10
-----------------------------------------------------
```

Zweck:

Eine Folge von Anweisungen soll auf Grund eines Ver-

gleichs ein- oder mehrmals ausgeführt werden.

Beschreibung:

Die Operation DO erlaubt es, eine Gruppe von Anweisungen (eine DO-Gruppe) einmal oder mehrere Male auszuführen. F1 kann ein numerisches Feld ohne Dezimalstellen sein, das den Anfangswert enthält, oder mit der Erweiterung 'UNTIL' oder 'WHILE' ein beliebiges numerisches oder alphanumerisches Feld oder Feldgruppenelement.

F2 kann ein numerisches Feld oder ein numerisches Literal ohne Dezimalstellen sein, das den Begrenzungswert enthält, oder mit der Erweiterung 'UNTIL' oder 'WHILE' ein beliebiges numerisches oder alphanumerisches Feld oder Feldgruppenelement.

Einschränkungen: die Feldart der beiden Faktoren muss übereinstimmen und bei alphanumerischen Faktoren darf nur ein Faktor indiziert sein.

In EG kann, wenn 'UNTIL' oder 'WHILE' nicht angegeben wurden, ein numerisches Feld ohne Dezimalstellen eingetragen werden, in das während der Ausführung der Operation der aktuelle Index abgestellt wird.

Ein Ergebnisfeld muss nicht spezifiziert werden. Fehlt das Ergebnisfeld, so steht der Index im Feld CPGDxx, wobei xx die Schachtelungsstufe angibt, die auch im Blockdiagramm der Umwandlungsliste angegeben ist.

Ist F2 nicht spezifiziert, ist der Begrenzungswert 1.

In SV kann das Schlüsselwort 'LOOP' eingetragen werden, wenn die Schleife endlos sein soll.

Wird bei einem DO UNTIL eine '1' in SV eingetragen, so wird die DO-Gruppe in jedem Fall mindestens ein mal ausgeführt.

Die zu verarbeitende Befehlsfolge muss durch ein 'END'- oder 'ENDDO'-Statement abgeschlossen werden. Dieses END kann um eine Bedingungsabfrage erweitert werden. Ist die Bedingung nicht erfüllt, wird das END-Statement ignoriert. Das hat zur Folge, dass die Schleifenverarbeitung beendet wird; das Programm wird vom Statement hinter dem END(DO) an weiter abgearbeitet.

Beachte: Korrektur der Indexvariablen in Release 1.6 !

Bei indizierten DO-Schleifen war das Indexfeld bis zum Release 1.5 nach Ausführung der Schleife mit einem Wert gefüllt, der größer als der Begrenzungswert war !

Beispiele dazu:

```
DO FROM 1 TO 12 WITH I. INDOF. ENDDO :  I ist nachher 12
DO FROM 1 TO 7 WITH J. INDOF. ENDDO 4 : J ist nachher  5
```

Tabelle der Operatoren :

Operator		Bedeutung		
GT	>	greater than	F1	größer F2
LT	<	less than	F1	kleiner F2
EQ	=	equal	F1	gleich F2
NE	><	not equal	F1	ungleich F2
GE	>=	greater equal	F1	größer oder gleich F2
LE	<=	less equal	F1	kleiner oder gleich F2

Die Operatoren UNTIL und WHILE für die Schleifensteuerung können analog zum IF um den Anhang -DAT, -DATI oder -DATK erweitert werden, also UNTIL-DAT,WHILE-DAT,UNTIL-DATI, WHILE-DATI etc.
Beschreibung siehe IF-DAT, IF-DATI oder IF-DATK.

DSPLY Ausgabe auf die Operator-Konsole 4195

```
(ON) F1 OP                                          online
(ON) (F1) OP (EG)                                    Batch
```

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Name des Feldes, das die Nachricht enthält
OP DSPLY oder DISPLAY muss eingetragen werden
EG Name des Feldes, das eine Nachricht/Antwort enthält

Beispiel :

```
NACHRICHT  DSPLY
NACHRICHT1 DISPLAY NACHRICHT2
NACHR      DISPLAY ANTWRT
```

Zweck:

Ausgabe von Nachrichten auf die Operator-Konsole. Bei Batchbetrieb auch Empfang von Antworten der Konsole.

Beschreibung:

Mit dieser Operation kann eine Nachricht auf der Operator-Konsole ausgegeben werden. Die Nachricht steht in einem alphanumerischen Feld, das maximal 255 Stellen groß sein darf (im Beispiel : NACHRICHT).

Im Batchbetrieb ist die Operation flexibler. Der Feldname für die Nachricht kann vor und/oder hinter dem Operationscode stehen. Werden zwei Nachrichten-Felder angegeben, so werden auch beide Feldinhalte auf der Konsole angezeigt.

Bei Batchbetrieb hält DSPLY das Programm an, bis es vom Operator wieder aktiviert wird. Eine 'Antwort' der Konsole wird im Ergebnisfeld an das Programm übergeben.

DUMP Dump ausgeben 4200

(ON) OP (F2)

ON Bedingungsabfrage (bis zu drei Schalter)
OP 'DUMP' muss eingetragen werden
F2 vierstelliger Dump-Code zur Unterscheidung

Beispiele:

- ON NOT DE DUMP
- DUMP NR01

Zweck:

Ausgabe eines CICS Transaction Dumps oder eines formatierten Batch-Dumps.

Beschreibung:

Diese Operation erzeugt einen CICS Transaction Dump auf der Platten-Einheit, die den CICS Dump File-Bereich enthält. Der Dump muss mit einem eigenen Programm gedruckt werden. Zur Unterscheidung verschiedener Dumps kann ein vierstelliger Dump-Code eingetragen werden.

Das Programm wird durch die Operation DUMP nicht beendet

EDIT alphanumerisches Feld aufbereiten 4205

```
-------------------------------------------------------
(ON) OP EG ((TYPE) F2) (SV) (BD)
-------------------------------------------------------
ON   Bedingungsabfrage (bis zu drei Schalter)
OP   'EDIT' muss eingetragen werden
EG   Name des aufzubereitenden Feldes
F2   Name der Edit-Ausgabe mit Schlüsselwort TYPe
SV   Service: 'INDex' : keine Indexprüfung
     oder:    'VAR' : variabler EDIT-Name (Typ)
BD   bis zu drei Schalter können gesetzt werden
-------------------------------------------------------
```

Beispiele:

```
- EDIT  FELD.
- EDIT  FG1(J) TYPE ABC.
- EDIT  FELD   TYPE OUT  VARIABEL.
```

Zweck:

Aufbereiten eines Feldes über die Output Division.

Beschreibung:

Mit dieser Operation können alphanumerische Felder aufbereitet werden. Es wird in die Output Division verzweigt, wo die Aufbereitung des Feldes beschrieben ist.

Die Operation EDIT ist indizierbar.

In Verbindung mit dem Schlüsselwort TYP kann eine bestimmte Ausgabe-Satzbestimmung gezielt angesprochen werden. Hinter TYP wird dazu der EDIT-Name direkt angegeben. Wenn es sich bei diesem EDIT-Namen um ein Feld handelt, dem zur Ausführungszeit der Name entnommen wird, gibt man zusätzlich das Service-Schlüsselwort VAR an. Das Feld muss sechstellig alphanumerisch definiert sein.

Der Service 'IND' bewirkt, dass der benutzte Index nicht geprüft wird. Der Programmierer trägt dann die Verantwortung dafür, dass kein Speicherbereich irrtümlich überschrieben wird.

Es kann nur eines der Schlüsselworte IND und VAR angegeben werden.

Es können bis zu drei Bezugszahlen gesetzt werden, die vor dem Verzweigen in die Output Division gesetzt werden und nach Ausführung der EDIT-Operation wieder gelöscht werden. Damit ist die Möglichkeit gegeben, ein Feld unter verschiedenen Bedingungen durch Schalter gesteuert unterschiedlich aufzubereiten.

ELIM(INATE) Zeichen löschen und durch Blank ersetzen 4210

```
-----------------------------------------------------
(ON) OP F2 EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   ELIM  oder ELIMINATE muss eingetragen werden
F2   zu entfernendes Zeichen als Literal/Hex-Konstante
EG   Alphafeld / -feldgruppe / -feldgruppenelement
-----------------------------------------------------

Beispiele:

-   ELIM '0'  FELD
-   ON PF17  ELIMINATE  X'00' INPUT
-----------------------------------------------------

Zweck:

Löschen eines Zeichens aus einem alphanumerischen Feld
und Ersetzen durch Blank.

Beschreibung:

Mit dieser Operation kann ein  Zeichen  in einem  alpha-
numerischen  Feld, in  einer  Feldgruppe  oder  in einem
Feldgruppenelement durch  Blank ersetzt  werden.  Das zu
ersetzende Zeichen wird entweder als einstelliges  Lite-
ral in  Hochkommata oder als  hexadezimale  Konstante in
der Form  X'00' angegeben.

Beispiel :   -  ELIM '*' FELD

Feldinhalt vorher  :        'A*B**C'
Feldinhalt nachher :        'A B  C'
```

ELSE Kennzeichnung eines Programmblocks im IF-Befehl 4215

```
-----------------------------------------------------
OP
-----------------------------------------------------

ELSE
-----------------------------------------------------

Die Operation ELSE kennzeichnet innerhalb einer IF-Grup-
pe den Programmzweig, der durchlaufen wird, wenn die Be-
dingung in der IF-Operation  nicht  erfüllt ist.
```

END Ende eines Programmblocks 4220

Die Operation END kennt in Abhängigkeit von der Operation, die sie abschließt, unterschiedliche Syntaxregeln. Es muss daher auf die Beschreibungen der 'Anfangs'operationen verwiesen werden.

END-EVALUATE Ende einer Mehrfachalternative 4222

```
------------------------------------------------------
OP   END-EVALUATE muss eingetragen werden
------------------------------------------------------
```

Die Operation END-EVALUATE schließt die mit EVALUATE eingeleitete Mehrfachalternative ab.

ENDDO Ende eines DO-Blocks 4225

```
------------------------------------------------------
OP (F2)
------------------------------------------------------
OP  ENDDO muss eingetragen werden
F2  Erhöhungswert konstant oder variabel
------------------------------------------------------
```

Die Operation ENDDO kann anstelle eines END als Schluss eines DO-Blocks (zum Beispiel zur besseren Dokumentation bei geschachtelten DO- und IF-Blöcken) benutzt werden.

In F2 kann konstant oder variabel ein numerischer Wert angegeben werden, um den die DO-Schleife nach jedem Durchlauf erhöht wird. Der Default-Wert ist 1.

ENDDT Ende einer Entscheidungstabelle 4230

```
------------------------------------------------------
OP  ENDDT muss eingetragen werden
------------------------------------------------------
```

Die Operation ENDDT schließt eine mit BEGDT eingeleitete Entscheidungstabelle ab (vgl. Kapitel 2410).

ENDEV Ende einer Mehrfachalternative 4232

OP ENDEV muss eingetragen werden

Die Operation ENDEV schließt die mit EVALUATE eingeleitete Mehrfachalternative ab.

ENDIF Ende eines IF-Blocks 4235

Die Operation ENDIF kann anstelle eines END als Schluss eines IF-Blocks (zum Beispiel zur besseren Dokumentation bei geschachtelten DO- und IF-Blöcken) benutzt werden. Unterschiede zum END bestehen nicht.

ENDSR Ende eines Unterprogramms 4240

```
-------------------------------------------------------
(F1)  OP
-------------------------------------------------------

F1   Anspringpunkt
OP   ENDSR  muss eingetragen werden
-------------------------------------------------------

Beispiele:

- ENDSR.
- LABEL5   ENDSR
-------------------------------------------------------

Zweck:

Beenden eines Unterprogramms.

Beschreibung:

Das Programm verzweigt bei dieser Operation  zurück zum
aufrufenden Befehl EXSR.
Es kann vor dem ENDSR ein  Label angelegt werden, z.B um
mit einem GOTO das  Unterprogrammende  direkt anspringen
zu können.
```

ENQ(UEUE) Programmteil sperren 4245

```
-------------------------------------------------------
OP F1 (SV)
-------------------------------------------------------

OP   ENQ  oder ENQUEUE muss eingetragen werden
F1   Name des Labels, bis zu dem gesperrt wird
SV   'EXTern' ( s.u., Beispiel )
-------------------------------------------------------

Beispiele:

- ENQ  A100.
- ENQ  C1  EXTERN
-------------------------------------------------------

Zweck:

Sperren eines Programmteils.
```

Beschreibung:

CPG-Programme sind reentrant, d.h. ein Programm kann zur gleichen Zeit von verschiedenen Terminals benutzt werden Insbesondere bedeutet dies, dass ein Plattensatz gleichzeitig von verschiedenen Stellen verändert werden kann.

Die Operation ENQ erlaubt dem Programmierer, zwischen dem Lesen und Zurückschreiben des Satzes die Benutzung eines Programmabschnitts solange zu sperren, bis der Update-Vorgang abgeschlossen ist.

Die Entriegelung erfolgt durch die Operation DEQ. (s.o.)

Durch den Befehl - ENQ X
wird das Programm von diesem Statement an für andere Benutzer solange gesperrt, bis der laufende Benutzer die Stelle X (d.h. den Befehl - X DEQ) erreicht hat.

Die Servicefunktion 'EXTern' bewirkt, dass auf einer bestimmten Stelle im Speicher, in der von allen Programmen erreichbaren CWA, die Information 'zur Zeit gesperrt' abgelegt und abgefragt werden kann.

Beispiel : Werden im User Copy Buch CPGUCCUA die Felder

```
C1        DS      C
C2        DS      C
C3        DS      C
```

definiert, kann im Programm ein Schutz auf Systemebene erreicht werden; auf einer 1-Byte großen Stelle C1 in der CSA (Common System Area) kann mittels ENQ C1 EXTERN die Information abgelegt werden, dass C1 - verriegelbare Programmteile zur Zeit gesperrt sind.

```
Programm 1:   -     ENQ  C1   EXT
                     :
              - C1  DEQ      EXT

Programm 2:   -     ENQ  C1   EXT
                     :
              - C1  DEQ      EXT
```

EREAD Bildschirmdialog (ersetzen durch MAP-Befehle !!!) 4250

(ON) OP F2 (SV) (BD)

ON Bedingungsabfrage (bis zu drei Schalter)
OP EREAD muss eingetragen werden
F2 Bildschirm-Datei
SV Service: CLEar, LOW, S, T, A, C, K
BD bis zu drei Schalter können gesetzt werden

Beispiele:

- EREAD BILD 20.
- EREAD BILD5 CLEAR

Zweck:

Ein Bildschirm wird vorformatisiert und eingelesen.

Beschreibung:

Diese Operation entspricht einem EXCPT mit nachfolgendem READ. Sind Ergebnisbezugszahlen angegeben, so werden sie vor der Verzweigung in die Output Division gesetzt und nach dem Einlesen des Bildschirms wieder gelöscht.

Die in der Output Division programmierten Felder werden zunächst auf den Bildschirm ausgegeben, das Programm erwartet eine Eingabe und liest nach Betätigung der Datenfreigabe oder einer anderen Programmfunktionstaste den Bildschirminhalt entsprechend den Vereinbarungen in der Input Division wieder ein.

Die Servicefunktionen:

'CLEar' Nach dem EREAD kann der Schalter CL (Löschtaste) abgefragt werden. Anderenfalls bewirkt die Löschtaste die Beendigung des Programms. Hardwarebedingt wird mit der Löschtaste grundsätzlich der Bildschirm gelöscht.

'LOW' Kleinbuchstaben werden nicht automatisch in Großbuchstaben übersetzt.

'S' Kombination von CLEAR und LOW

Für 'quasi-transaktionsorientiertes' Programmieren stehen vier weitere Services zur Verfügung:

T für die transaktionsorientierte Programmierweise
A impliziert T und LOW
C impliziert T und CLEar
K impliziert T, LOW und CLEar

Achtung: Groß-/Kleinschreibung ist beim transaktionsorientierten Programmieren nur möglich, wenn in der TCT der Parameter UCTRAN nicht definiert wurde bzw. wenn im Programm der Befehl UCTRAN OFF codiert wurde; außerdem muss in der OPTIONS-Karte 'LOW' eingetragen sein.

Hinweis: Diese 'quasi-transaktionsorientierte' Programmierweise unterliegt einigen Einschränkungen. Sie darf nicht eingesetzt werden
- in Programmen, die mit EXPR oder EXITP mit Programmnamen aufgerufen werden
- in HL1-Bausteinen
- wenn ein Dataset im Programm definiert ist
- wenn ein CHAIN für Update über der Operation liegt (CHAIN U wird dann ignoriert).

Sie sollte nur nach ausreichender Schulung angewandt werden.

EVALUATE Mehrfachalternative 4255

OP EVALUATE muss eingetragen werden

Zweck:

Die EVALUATE-Operation wird eingesetzt, wenn (höchstens) eine von mehreren Alternativen ausgeführt werden soll.

Beschreibung:

Als Operationsbezeichnung muss EVALUATE angegeben werden. Die Alternativen werden mit WHEN in den Folgezeilen gekennzeichnet. Die Faktoren für die Bedingungen können numerische oder alphanumerische Felder, Feldnamen oder Feldgruppenelemente enthalten. Ist eine Bedingung erfüllt, werden die nachfolgenden Anweisungen bearbeitet. Danach ist die EVALUATE-Anweisung beendet und das Programm wird hinter dem END-EVALUATE fortgesetzt.

Die Bedingung WHEN OTHER ist erfüllt, wenn keine der vorhergehenden WHEN-Anweisungen zutreffend war. Die zugehörigen Anweisungen werden in diesem Fall ausgeführt und das EVALUATE ist beendet.

(Siehe auch Beispiele 24 und 25 in Kapitel 8000)

Beachte:

In einer EVALUATE-Statementgruppe sollte BREAK nicht eingesetzt werden, weil EVALUATE eine eigene DO-Stufe darstellt. Liegt also eine EVALUATE-Gruppe in einer DO-Schleife, so verzweigt das BREAK innerhalb der Gruppe (nur) hinter das END-EVALUATE und nicht hinter das folgende ENDDO.

EXCPT Ausgabe 4260

```
-----------------------------------------------------
(ON) OP (F2) (SV) (BD)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXCPT muss eingetragen werden
F2   2- bis 30-stelliger Name oder 6-stelliges Feld
SV   VARiabel ( in Verbindung mit Feldname in F2 )
BD   bis zu drei Schalter können gesetzt werden
-----------------------------------------------------
```

Beispiele:

```
- EXCPT.                * alles ausgeben
- EXCPT KUNDEN-AENDERN. * zu KUNDEN-AENDERN verzweigen
- EXCPT 01 11.          * 01 / 02 setzen, in die Ausgabe
                        * verzweigen, 01/02 löschen
- EXCPT FELD VARIABEL.  * der EXCPT-Name steht in FELD
-----------------------------------------------------
```

Zweck:

Die Ausgabe soll ausgeführt werden.

Beschreibung:

Diese Operation entspricht einer Verzweigung in die Output-Division. Im einfachsten Fall (- EXCPT.) werden die Ausgaben für alle Dateien ausgeführt.

Enthält die EXCPT-Anweisung einen Namen oder Bezugszahlen, so werden die Ausgaben ausgeführt, die mit diesem Namen oder diesen Bezugszahlen verriegelt sind.

Beachte:

Es werden immer auch alle Ausgaben ausgeführt, die nicht verriegelt sind.

Beispiele:

```
Rechenbestimmung:  -  EXCPT ASATZ
:
Ausgabe:           - -O.  FILE ARTIKEL ASATZ
```

Werden eine oder mehrere Bedingungen angegeben, so werden diese vor Ausführung der Operation gesetzt und nach Ausführung wieder gelöscht.

```
Rechenbestimmung:  -      EXCPT 25
:
Ausgabe:           - -O.  FILE ARTIKEL ON 25
```

EXECUTE Anderes Programm ausführen 4265

identisch mit EXPR (s.u.)

EXHM HL1-Modul ausführen (ab Service-Level CPG3) 4270

```
-----------------------------------------------------------
(ON) OP F2 (EG) (SV)
-----------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXHM  muss eingetragen werden
F2   Name eines HL1-Moduls oder eines 8-stelligen Feldes
EG   Name eines HL1-Datenkanals
SV   I  : Baugruppe initialisieren,  T  : I + Pf-Tasten
     V  : variables EXHM (ohne Angabe des Kanals)
-----------------------------------------------------------

Beispiele:

- EXHM  HB0001
- EXHM  HB0001  KANAL1
- EXHM XHMFLD VAR.
-----------------------------------------------------------

Zweck:

Ein HL1-Baustein soll ausgeführt werden.

Beschreibung:

In F2 wird der Name eines Bausteins, der in der  HL1-Ta-
belle enthalten sein muss, angegeben oder ein  achtstel-
liges Feld, das bei  variabler Verarbeitung diesen Namen
in den  ersten sechs Stellen  enthält. In der  Stelle 7
steht die private HL1-Library, Stelle 8 ist frei.

EG kann den  Namen eines  Datenkanals  enthalten, der in
der  Input-Division beschrieben sein muss. Beim Aufruf
des Bausteins werden dann alle unter dem  Datenkanal be-
schriebenen Felder in die private TWA ( Transaction Work
Area ) des Bausteins übertragen;  nach dessen  Ausfüh-
rung werden die  veränderten  Feldinhalte wieder in die
entsprechenden Felder des aufrufenden Programms zurück-
übertragen.

Bei variabler Verarbeitung ist diese Form  der  Daten-
übergabe nicht unterstützt. Sollen  bei  variablem  EXHM
Daten zwischen rufendem und gerufenem Modul ausgetauscht
werden, so muss sich das aufgerufene Modul die Daten beim
rufenden abholen. Dazu muss ein Eintrag SHA für 'Shared
Data' in den  Options des  gerufenen  Programms  gesetzt
werden. Dann werden Daten zwischen  Feldern mit gleichem
Namen und gleichen Feldeigenschaften ausgetauscht.

Die Servicefunktion 'V'  beinhaltet die  Servicefunktion
'T', die im folgenden erklärt ist.

Mit der Funktion 'I' oder INIT bewirkt man,dass das auf-
rufende Programm für die somit initialisierte Baugruppe
```

den Status eines Hauptprogramms erhält. Das bedeutet, dass die Operationen GETMI und PUTMI sowie EDIT CPGTWA und SELCT CPGTWA sich für alle Bausteine auf einer der folgenden Stufen auf die Bezugszahlen bzw. die TWA des Programms beziehen, in dem die Baugruppe initialisiert wurde.

Bei der Betätigung der Löschtaste einer Bildschirmtastatur in einem untergeordneten Baustein wird in diesem Fall (Service 'I') die Verarbeitung mit der Operation, die der EXHM-Operation folgt, fortgesetzt.

Service 'T' beinhaltet Service 'I'. Zusätzlich können Programmfunktionstasten abgefragt werden.

EXHM-VAR HL1-Modul ausführen (ab Service-Level CPG3) 4273

(ON) OP F2

OP EXHM-VAR muss eingetragen werden
F2 Name eines achtstelligen Alphafeldes

Beispiel:

- EXHM-VAR XHMFLD

Zweck:

Ein HL1-Baustein soll ausgeführt werden.

Beschreibung:

In F2 wird der Name eines achtstelligen Feldes angegeben. Dieses enthält in den ersten sechs Stellen den Namen des auszuführenden Bausteins und in Stelle 7 gegebenenfalls die private HL1-Library. Stelle 8 ist frei.

Sollen bei variablem EXHM Daten zwischen rufendem und gerufenem Modul ausgetauscht werden, so muss sich das aufgerufene Modul die Daten beim rufenden abholen.

Dazu muss ein Eintrag SHA für 'Shared Data' in den Options des gerufenen Programms gesetzt werden. Dann werden Daten zwischen Feldern mit gleichem Namen und gleichen Feldeigenschaften ausgetauscht.

EXITD Anderes Programm aufrufen mit Datenübergabe 4275

```
-------------------------------------------------------
(ON) OP EG (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXITD muss eingetragen werden
EG   Name einer Datenstruktur
SV   'T' für Uhrzeit ( statt Zeitintervall )
-------------------------------------------------------
```

Beispiel:

```
- EXITD  STRUKT
-------------------------------------------------------
```

Zweck:

Aufrufen eines anderen Programms mit Datenübergabe.

Beschreibung:

Die Operation ist eine Erweiterung der Operation EXITI.

Ein grundsätzlicher Unterschied besteht darin, dass bei EXITD das Programm nicht verlassen wird.

In EG wird der Name einer Datenstruktur eingetragen, die in der Input-Division wie folgt beschrieben sein muss:

```
- I. FIELD  STRUKT  DS.
-     1  4  TRANID   Trans-Id der Folgetask
-     5  8  TERMID   Terminal-Id, an dem die Folgetask
                     gestartet wird. Default : gleiches
                     Terminal
- PAC 9 12 0 ZEIT    Entweder Uhrzeit oder Intervall zum
                     Zeitpunkt des Befehls. Gibt an,wann
                     die Folgetask gestartet werden soll
-    13 20  TSNAME   Temporary Storage Name, wird, falls
                     Blank, vom TP-Monitor vergeben.
- *  21 24  INTERN   wird intern vom CPG gefüllt
-    25 88  DATEN    zu übergebende Daten. Die Länge
                     des Datenbereichs kann natürlich
                     höher gewählt werden.
```

Die Folgetask kann die übergebenen Daten über eine READ-Operation auf "$CPG", einer CPG-internen Temporary Storage Queue, lesen. Sind keine Daten vorhanden, dann wird der Schalter EF gesetzt. Wenn kein READ auf $CPG erfolgt, so werden diese Daten automatisch beim Ende der Task gelöscht.

Die Service-Funktion 'T' gibt an, dass in den Stellen 9 bis 12 der Datenstruktur eine feste Uhrzeit in der Form 0HHMMSSC übergeben wird. Die Alternative ist ein Zeitintervall vom EXITD-Befehl an, etwa 20 für einen Start der Folgetask nach 20 Sekunden oder 230 für ein Intervall von 2 Minuten und 30 Sekunden.

EF wird gesetzt, wenn entweder die auszuführende Task oder das angesprochene Terminal nicht in den entsprechenden CICS-Tabellen definiert ist. (nicht bei Macro Level!)

Die Folgetask kann die übergebenen Daten wie folgt lesen:

```
- -I. FILE $CPG.      1  54 DATEN
- -C.        READ $CPG
-            IF CONDITION NOT EOF.  .....
```

Unter $CPG wird eine Pseudo-TS-Queue gelesen; ist diese nicht vorhanden, wird der Schalter EF gesetzt.

EXITI Anderes Programm über Intervall Control aufrufen 4280

(ON) OP F2

ON Bedingungsabfrage (bis zu drei Schalter)
OP EXITI muss eingetragen werden
F2 Transaktion in Hochkommata oder als Variable

Beispiele:

- EXITI 'PG02'
- ON PF4 EXITI TRID

Zweck:

Aufrufen eines anderen Programms.

Beschreibung:

F2 enthält die Transaktion des aufzurufenden Programms entweder als vierstellige Konstante in Hochkommata oder als variables Alphafeld. Im zweiten Fall ist der Programmierer dafür verantwortlich, dass das vierstellig definierte Feld zur Zeit des EXITI eine gültige Transaktion enthält.

EXITI verzweigt sofort in das Folgeprogramm; dieses liest beim Start keine Daten vom Bildschirm ein.

Hinweis:

Erweiterte Möglichkeiten wie zeitverzögertes Starten einer Task auch an einem anderen Bildschirm bietet die Operation EXITS.

EF wird gesetzt, wenn die aufgerufene Task nicht in der der PCT liegt (nicht bei Macro Level!).Siehe auch EXITS.

EXITP Anderes Programm aufrufen 4285

```
-----------------------------------------------------
(ON) OP F2 (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXITP muss eingetragen werden
F2   Transaktion in Hochkommata  oder  Programmname
SV   'SAVe' zum Zwischenspeichern der TWA
-----------------------------------------------------
```

Beispiele:

- EXITP TST024
- EXITP 'PA36' SAVE

Zweck:

Aufrufen eines anderen Programms.

Beschreibung:

Diese Operation ruft ein anderes in der Programm-Tabelle des TP-Monitors definiertes Programm auf.

1. Möglichkeit

In F2 steht der Phasenname des Programms, in das verzweigt werden soll, als Konstante (ohne Hochkommata). In diesem Fall bleibt die Task erhalten, lediglich das Programm wird ausgetauscht.

Sollen Daten übernommen werden, so sind diese in den Data-Divisions beider Programme vom ersten definierten Feld an deckungsgleich zu beschreiben.

Es ist darauf zu achten, dass keine Schleifen entstehen, da diese zu erheblichen Performance-Verlusten führen können.

Diese Operation entspricht dem EXEC CICS XCTL.

Die Service-Funktion 'SAVe' bewirkt, dass die TWA auf eine CPG-interne Temporary Storage Queue zwischengespeichert und im Folgeprogramm in der definierten Länge übernommen wird. (Vgl. dazu den Parameter TWA xxxx in der OPTIONS-Bestimmung.) Dieses Save ist dann wichtig, wenn in einem Taskzyklus mehrere Programme mit EXPR sowie EXITP (mit Programmname) verknüpft sind.

Siehe dazu auch die Beschreibung unter EXPR.

Die Operation EXITP mit Programmnamen schließt eine RNDOM*ALL Operation ein, außer wenn in den Options der Parameter USE angegeben wurde.

Bei Programmverbindungen (EXITP, EXPR) in ESA-Umgebungen

größer 16 MB müssen alle verbundenen Programme auf der gleichen Seite der 16 MB-Linie liegen !

2. Möglichkeit

In F2 kann auch in Hochkommata eingeschlossen die Trans-Id einer Folgetask gesetzt werden. Nach Betätigen einer Programmfunktionstaste wird dann diese Task gestartet. Im Wesentlichen entspricht diese Funktion der Operation EXITT mit dem Unterschied, dass MDT-Bits am Bildschirm gelöscht werden.

EF wird gesetzt, wenn das aufgerufene Programm nicht in der PPT liegt (nicht bei Macro Level!).

EXITP-VAR Anderes Programm variabel aufrufen 4290

```
-------------------------------------------------------
(ON) OP EG (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXITP-VAR muss eingetragen werden
EG   Name eines vier- oder achtstelligen Alphafeldes
SV   'SAVe' zum Zwischenspeichern der TWA
-------------------------------------------------------
```

Beispiele:

- EXITP-VAR PROGN
- EXITP-VAR TRID SAVE

Zweck:

Aufruf eines anderen Programms (siehe EXITP)

EXITS Programm (später) an anderes Terminal schicken 4295

```
-------------------------------------------------------
(ON) (F1) OP F2 EG (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Terminal in Hochkommata oder als Variable
OP   EXITS oder EXIT-SEND muss eingetragen werden
F2   Transaktion in Hochkommata oder als Variable
EG   Zeitverzögerung als Konstante oder Variable
SV   'T': EG enthält Uhrzeit, 'N': Prog nicht verlassen
-------------------------------------------------------

Beispiele:

-        EXITS     TRID   #
- 'NONE' EXIT-SEND 'TR01' '0005'  N
- 'DV14' EXITS     TRID   ZEIT    T
-  TERM  EXIT-SEND 'TR02' #       N
-------------------------------------------------------

Zweck:

Aufrufen eines anderen Programms an einem  anderen Bild-
schirm und/oder zeitverzögert.

Beschreibung:

F2 enthält die Transaktion des  aufzurufenden Programms
entweder als vierstellige Konstante in Hochkommata  oder
als variables Alphafeld. Im  zweiten  Fall ist der  Pro-
grammierer dafür verantwortlich,  dass das  vierstellig
definierte Feld zur Zeit des EXITS eine  gültige Trans-
aktion enthält.

F1  enthält die  Terminal-Id eines  Bildschirms, an dem
die in F2 angegebene Transaktion  gestartet werden soll,
entweder in Hochkommata  oder als  vierstelliges  Alpha-
feld. Im zweiten Fall muss der Programmierer sicherstel-
len, dass das Feld zur Zeit des  EXITS mit einer gülti-
gen Trans-Id gefüllt ist.

Zum Start einer Non-Terminal Task wird 'NONE' in F1 ein-
getragen.

In EG wird die Zeitverzögerung entweder als vierstelli-
ge Konstante in Hochkommata oder als 7-stelliges numeri-
sches Feld mit Inhalt '0HHMMSS' angegeben. Eine Konstan-
te wird interpretiert als 'HHMM'.

Beachte: Wird keine Zeitverzögerung gewünscht, so muss
         stattdessen ein '#' eingetragen werden.

Per  Servicefunktion wird unterschieden,  ob es sich bei
bei der  Zeitangabe in EG um eine  feste Uhrzeit oder um
eine Verzögerungszeitspanne handelt.  Wird  Service 'T'
angegeben, so wird  EG als  Zeitpunkt  interpretiert, an
dem die Transaktion gestartet wird,  ansonsten als Zeit-
```

intervall bis zum Start.

Beispiel: 'DV01' EXIT-SEND TRID '0005'

TRID wird am Terminal DV01 nach 5 Minuten gestartet.

EXITS verzweigt sofort in das Folgeprogramm; dieses liest beim Start keine Daten vom Bildschirm ein.

Wird jedoch als Service ein 'N' eingetragen, so wird das laufende Programm durch das EXIT-SEND nicht verlassen. Das aufgerufene Programm wird also erst dann ausgeführt wenn das laufende Programm beendet ist.

Der Service 'S' schließt 'T' und 'N' ein.

EF wird gesetzt, wenn die aufgerufene Task nicht in der PCT liegt (nicht bei Macro Level!).

EXITT Anderes Programm aufrufen mit Bildschirmdaten 4300

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP EXITT oder EXIT-TRANS muss eingetragen werden
F2 Transaktion in Hochkommata (vierstellig)
SV VAR, falls mit variabler Trans-Id gearbeitet wird

Beispiel:

- EXITT 'TST1'
- EXITT TRID VARIABLE

Zweck:

Aufruf eines anderen Programms (transaktionsorientiert)

Beschreibung:

Die Operation EXITT arbeitet wie die EXITP-Operation. Der Unterschied besteht darin, dass die Bildschirmtastatur nach Ausführung der Operation nicht entriegelt wird und vom Programm gesetzte Modified Data Tags nicht gelöscht werden.

Diese Operation ist immer dann zu verwenden, wenn bei READ-freier (taskorientierter) Programmierung Daten auf dem Bildschirm zwischengespeichert bzw. gelesen werden sollen.

EF wird gesetzt, wenn die aufgerufene Task nicht in der PCT liegt (nicht bei Macro Level!).

EXITT-VAR variables EXITT 4305

(ON) OP EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP EXITT-VAR muss eingetragen werden
EG Transaktion in vierstelligem Alphafeld

Beispiel:

- MOVE 'TST1' TRID
- EXITT-VAR TRID

Zweck:

Aufruf eines anderen Programms (transaktionsorientiert)

EXIT-SEND Programm (später) an anderes Terminal schicken 4310

---

identisch mit EXITS (s.o.)

EXIT-TRANS Anderes Programm aufrufen mit Bildschirmdaten 4315

---

identisch mit EXITT (s.o.)

EXPR Anderes Programm ausführen 4320

```
-----------------------------------------------------
(ON) (F1) OP F2 (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Numerisches Feld für Länge der Common Area
OP   EXPR  oder  EXECUTE  muss eingetragen werden
F2   Phasenname als Konstante ( ohne Hochkommata )
SV   'IND', 'SAVe'
-----------------------------------------------------

Beispiele:

- EXPR  TST001  SAVE
- EXPR  PROG44
-----------------------------------------------------

Zweck:

Aufrufen eines anderen Programms.

Beschreibung:

Mit diesem Befehl kann an  dieser  Stelle des  Programms
ein anderes CPG- (oder CICS-) Programm  ausgeführt wer-
den. Nach Beendigung der Ausführung  wird die Verarbei-
tung mit der nächsten Instruktion fortgesetzt.

Voraussetzung ist, dass die TWAs des aufrufenden und des
aufgerufenen  Programms  übereinstimmen.  F2  enthält
den Phasennamen des auszuführenden Programms.

Es ist darauf zu achten, dass keine Schleifen entstehen,
da diese zu erheblichen Performance-Verlusten führen
können. Diese Operation entspricht dem EXEC CICS LINK.

Die Service-Funktion 'SAVe' rettet die TWA des aufrufen-
den Programms auf Temporary Storage in eine CPG-interne
Temporary Storage Queue  ("TERM$CPG").  Diese sollte vom
Anwendungsprogrammierer nicht verwendet werden.

Beispiel: Bezugszahlen bei Service 'SAVE'

-  EXPR  PROG2  SAVE.       * Befehl im Programm PROG1

In  PROG1 sind die Schalter 01 und 02 gesetzt.  Nach dem
EXPR sind 01 und 02  auch in  PROG2 gesetzt. Schalter 99
wird in PROG2 gesetzt.  Nach dem Rücksprung in PROG1 ist
dort der  gleiche Zustand  wie  vor dem EXPR:  01 und 02
sind gesetzt,  99  nicht.
Wird allerdings in der  Options Division  von  PROG2 der
Parameter TWA und die Anzahl zu  übernehmender  Stellen
angegeben, so werden  alle in  PROG2  gesetzten Schalter
beim Rücksprung nach PROG1 übergeben; im Beispiel wären
also  01, 02  und 99 gesetzt.

In der OPTIONS-Bestimmung wird über den  Parameter  TWA
```

bestimmt, wieviele Stellen aus der Transaction Work Area übernommen werden.

Kehrt die Kontrolle in das rufende Programm zurück, so wird die gleiche Anzahl in die TWA des aufrufenden Programms zurück übertragen, die zuvor übernommen wurde. Die Bezugszahlen werden ebenfalls aus dem aufgerufenen Programm übernommen. Der Rest der TWA hat den alten Inhalt, also genau so wie vor dem Aufruf des Unterprogramms.

Die Service-Funktion 'INDicator' entspricht dem 'SAVe' mit dem Unterschied, dass die Bezugszahlen nicht aus dem Unterprogramm übernommen werden, sondern von Temporary Storage.

Die Operation EXPR mit Programmnamen schließt eine RNDOM*ALL Operation ein, außer wenn in den Options der Parameter USE angegeben wurde.

EF wird gesetzt, wenn das aufgerufene Programm nicht in der PPT liegt (nicht bei Macro Level!).

Optional kann vor dem EXPR ein numerisches Feld ohne Dezimalstellen angegeben werden. Der Inhalt dieses Feldes bestimmt bei der Ausführung des EXPR die Länge, in der die Common Area verarbeitet wird (maximal 4080 Bytes).

Bei Programmverbindungen (EXITP, EXPR) in ESA-Umgebungen größer 16 MB müssen alle verbundenen Programme auf der gleichen Seite der 16 MB-Linie liegen !

EXPR-VAR Anderes Programm ausführen 4325

```
-----------------------------------------------------
(ON) OP EG (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXPR-VAR  muss eingetragen werden
EG   Phasenname in achtstelligem Alphafeld
SV   'IND', 'SAVe'
-----------------------------------------------------

Beispiele:

- MOVE-LEFT 'PHASE'  TO PROGN
- EXPR-VAR  PROGN
- EXPR-VAR  PROGN  SAVE
-----------------------------------------------------

Zweck:

Aufrufen eines anderen Programms ( wie EXPR )
```

EXSR Unterprogramm ausführen 4330

```
-----------------------------------------------------
(ON) OP F2 (BD)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   EXSR  oder PERFORM  muss eingetragen werden
F2   Name eines Unterprogramms
BD   bis zu drei Schalter können gesetzt werden
-----------------------------------------------------

Beispiele:

- EXSR  UP1
- PERFORM  DATUMSPRÜFUNG
-----------------------------------------------------

Zweck:

Ein Unterprogramm soll ausgeführt werden.

Beschreibung:

Mit der Operation  EXSR verzweigt das Programm in die in
F2 angegebene  Unterroutine, die am  Ende der  Procedure
Division programmiert sein muss.

Es können bis zu drei Schalter angegeben werden;  diese
werden dann vor dem Unterprogrammsprung gesetzt und nach
dem Rücksprung wieder gelöscht.
```

FILL Feld mit einem Zeichen füllen 4335

(ON) OP F2 (DY) EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP FILL muss eingetragen werden
F2 Zeichen in Hochkommata oder Hex-Konstante
EG Name eines alphanumerischen Feldes

Beispiele:

- FILL ' ' INTO PAGE
- FILL X'00' LINE24

Zweck:

Ein alphanumerisches Feld oder eine Datenstruktur sollen mit einem Zeichen gefüllt werden.

Beschreibung:

Mit der Operation FILL kann ein alphanumerisches Feld mit jedem beliebigen darstellbaren und nicht darstellbaren Zeichen gefüllt werden.

F2 enthält das Füllzeichen entweder als einstellige Konstante in Hochkommata oder als hexadezimale Konstante in der Form X' ' (vgl. Beispiele).

Beispiel: - FILL '*' FELD

Feldinhalt vorher (123)
Feldinhalt nachher (******)

FIND Durchsuchen einer Datenview 4340

(ON) F1 OP F2 (BD)

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Suchargument
OP FIND muss eingetragen werden
F2 (vierstelliger) Name der Tabelle
BD Schalter für Abfrage auf '=' in der Form '# # BD'

Beispiel:

- K FIND TAB1

Zweck:

Eine (extern erstellte) Datenview wird nach einem Feldinhalt durchsucht.

Beschreibung:

Voraussetzung ist, dass eine Tabelle angelegt wurde, die mit FIND verarbeitet werden kann. Zum Erstellen einer solchen Tabelle und Regeln, die bei der Verarbeitung von Datenviews zu beachten sind, vergleiche Kapitel 2340.

F1 enthält den Namen eines Feldes, das die zu durchsuchende Spalte der Tabelle angibt und nach dessen Inhalt diese Spalte durchsucht wird.

In F2 steht der vierstellige Name der Tabelle.

Wird das Suchargument nicht gefunden, so wird CPGFRC auf 'EF' gesetzt.

Es kann eine Bezugszahl in der Form '# # BD' angegeben werden, die gesetzt wird, wenn das Suchargument gefunden wurde. Ansonsten wird EF gesetzt, man kann also auf den Schalter verzichten.

GETCHANNEL	Kanal erneut übertragen	4342
GETHS	Kanal erneut übertragen	4342

(ON) OP

ON Bedingungsabfrage (bis zu drei Schalter)
OP GETCHANNEL oder GETHS muss eingetragen werden

Zweck:

In einem HL1-Baustein soll der Datenkanal so wiederhergestellt werden, wie er beim Aufruf des Datasetmoduls gefüllt war.

Beschreibung:

GETCHANNEL wird z.B. in HL1-Datasets eingesetzt. Bei der Programmierung einer logischen Datei in einem Dataset kann die Schwierigkeit auftreten, dass Kanalfelder durch Lese-Operationen überschrieben wurden, z.B beim CHAIN vor einem Update.

Somit ist es möglich, mit einem HL1-Dataset die Update-Funktion auch dann zu realisieren, wenn nur ein Teil der Felder im Datenkanal übertragen wird. (Siehe Beispiel 1 : 1 Dataset im HL1-Handbuch)

Die Operation setzt jeweils alle Kanal-Felder auf den Wert zurück, den sie bei Aufruf des Datasets hatten.

GETIN Schalter der nächsthöheren Stufe abfragen (HL1) 4345

```
-----------------------------------------------------
(ON) OP BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   GETIN muss eingetragen werden
BD   bis zu drei Schalter
-----------------------------------------------------
```

Beispiel :

```
-  GETIN  90 10
-----------------------------------------------------
```

Zweck:

Zustand einer Bezugszahl in ein Programm übernehmen.

Beschreibung:

Mit GETIN kann eine Bezugszahl von der nächst höheren HL1-Stufe ins Programm geholt werden. Dabei wird der Zustand der in BD angegebenen Bezugszahl(en) von der höheren Stufe in die gleichlautende(n) Bezugszahl(en) des aufrufenden Programms übertragen.

Der vorherige Zustand der Schalter im aufrufenden Programm bleibt unberücksichtigt; er wird überschrieben.

GETMI Schalter der höchsten HL1-Stufe abfragen 4350

```
-----------------------------------------------------
(ON) OP BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   GETMI muss eingetragen werden
BD   bis zu drei Schalter
-----------------------------------------------------
```

Beispiel :

```
-  GETMI  90 10
-----------------------------------------------------
```

Zweck:

Zustand einer Bezugszahl in ein Programm übernehmen.

Beschreibung:

Mit GETMI kann eine Bezugszahl von der nächst höheren HL1-Hauptprogrammstufe ins Programm geholt werden (vgl. GETIN).

GET-UPDATE Datensatz direkt lesen 4355

identisch mit CHAIN (s.o.)

GO(TO) Verzweigen nach 4360

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP GOTO oder GO muss eingetragen werden
F2 Anspringpunkt
SV E ermöglicht das Verzweigen in Subroutines

Beispiele:

- GOTO ENDE.

Zweck:

Die Verarbeitung soll an anderer Stelle fortgesetzt werden.

Beschreibung:

Diese Operation verzweigt zu dem in F2 genannten Merkmal. Dieses Merkmal muss mit Hilfe einer 'TAG'- Operation definiert werden.

```
- ON PF1 GOTO NEXT.
  ...
- NEXT.              * TAG kann weggelassen werden !!
```

IF Wenn - Abfrage 4365

(ON) OP F1 (DY) OK F2 (DY) (BO)

oder

OP SV BD (BD) (BD)

ON Bedingungsabfrage (bis zu drei Schalter)
OP 'IF' muss eingetragen werden
F1 erstes Vergleichsfeld (indizierbar)
OK Operator, vgl. unten : Tabelle der Operatoren
F2 zweites Vergleichsfeld (indizierbar)
DY Dummyworte (wie z.B. IS, THEN)
BO Boolesche Operatoren AND oder OR

oder

OP 'IF' muss eingetragen werden
SV Schlüsselwort CONdition (Bezugszahlenabfrage)
BD Schalter

Beispiele:

```
- IF A = B.                          * wenn A = B
- IF A GT B.                         * wenn A > B
- IF SUMME GREATER  FG(IND)
- IF A IS NOT GREATER THAN B THEN
- IF A = 0 OR
- IF B = 0
```

oder

```
- IF CONDITION  CLEAR
- IF CON  PF12.
```

Zweck:

Ein oder zwei Programmabschnitte sollen auf Grund eines Vergleichs ausgeführt oder nicht ausgeführt werden.

Beschreibung:

Die IF-Operation ermöglicht die Ausführung einer Gruppe von Anweisungen unter der Bedingung, dass eine vereinbarte Beziehung zwischen F1 und F2 besteht. Die Gruppe wird durch ein 'END(IF)'- Statement abgeschlossen.

```
- IF A = B. PI = 3,14. X = D * PI. END.
```

Falls gewünscht, kann auch ein Alternativzweig durchlaufen werden, wenn die o. g. Beziehung nicht besteht. Dies wird durch die Operation 'ELSE' bewirkt, die wahlweise zwischen 'IF' und 'END' Statement eingefügt werden kann und den IF-Zweig vom ELSE-Zweig trennt.

```
-   IF M > 6.  X = A * 12.
-   ELSE.      X = A * 6.
-   END
```

Tabelle der Operatoren :

Operator			Bedeutung	
GT	>	GREATER THAN	F1	größer F2
LT	<	LESS THAN	F1	kleiner F2
EQ	=	EQUAL	F1	gleich F2
NE	><	NOT EQUAL	F1	ungleich F2
GE	>=	GREATER EQUAL	F1	größer oder gleich F2
LE	<=	LESS EQUAL	F1	kleiner oder gleich F2

IF ist in beiden Faktoren indizierbar.

Die gleiche Verarbeitungslogik ist auch in Abhängigkeit vom Zustand eines oder mehrerer Schalter unterstützt.

```
-   IF CON DE. KEY UPDAT DATEI SATZ
-   ELSE.          DELET DATEI
-   ENDIF
```

oder

```
- IF CONDITION  99 AND NOT 10. EXSR FEHLER.
- ELSE.                        EXSR KALKULATION
- ENDIF.
```

Es können die Zustände von bis zu drei Schaltern abgefragt werden. Die angegebenen Bedingungen sind logisch und-verknüpft; im letzten Beispiel wird also das Unterprogramm 'FEHLER' ausgeführt, wenn Schalter 99 gesetzt und Schalter 10 nicht gesetzt ist. Ist eine der Bedingungen nicht erfüllt, wird das Unterprogramm 'KALKULATION' ausgeführt.

Werden mehrere Bezugszahlen abgefragt, so können im Befehl die Dummyworte AND und UND zur Dokumentation der logischen Verknüpfung angegeben werden.

Für die Negation (Schalter ist aus) sind die Schlüsselworte NOT und NICHT vorgesehen.

In Verbindung mit den booleschen Operatoren AND und OR gelten folgende Syntaxregeln:

- Zu einer logischen Verknüpfung von IFs gehört immer nur ein END(IF).
- OR und AND stehen immer hinter dem IF-Statement
- mehrere IF pro Zeile können codiert werden (Ausnahme von der Regel: Nur ein Befehl pro Statement)
- Ab der zweiten Bedingung kann das Schlüsselwort IF weggelassen werden.

Beispiele:

```
-  IF A = B  AND.              *  Und-Verknüpfung
-  IF B > C
-     EXSR SR03
-  END

-  IF A = 0  OR.               *  Oder-Verknüpfung
-     B = 0  OR
-     C = 0
-     EXSR SR03
-  END

-  IF KDNR > '     '  OR  IF SUMME > 0  OR  IF X = 0
-     EXSR SR04
-  ENDIF

-  IF  A > B  AND  IF A > C   OR
-  IF  A = B  AND  IF C = 0
-      EXSR  SR05
-  ENDIF
```

AND und OR können gemischt verwendet werden. Es gelten dabei die Regeln der mathematischen Aussagenlogik. Verkürzt: AND bindet stärker als OR. Somit ist das Beispiel oben eindeutig. SR05 wird ausgeführt, wenn eine der beiden Und-Verknüpfungen wahr ist.

IFDAT Datumsfelder vergleichen, Format ttmmjj 4368

OP F1 OK F2 (BO)

OP IF-DAT oder IF-DATE
F1 Name eines Datumsfeldes
OK Vergleichsoperator (siehe IF)
F2 Name eines Datumsfeldes
BO logische Verknüpfung mit AND und OR

Beispiele:

```
IF-DAT LOEDAT > UDATE
IF-DAT ALPHA8 < NUM20

C           LOEDAT    IFGT UDATE                D
C           ALPHA8    IFLT NUM20                D
```

Zweck:

Vergleich von Datumsfeldern, die in der Form ttmmjj (Tag,Monat,Jahr) gefüllt sind. Durch den Einsatz von IF-DAT spart man die Konvertierung der Felder vor dem Vergleich.

Beschreibung:

IF-DAT vergleicht intern die Felder CPGWD1 und CPGWD2, die das Datum achtstellig alphanumerisch im ISO-Format enthalten.

IF-DAT kann alphanumerische und numerische Felder miteinander vergleichen, auch wenn Länge und/oder Typ unterschiedlich sind. Die Vergleichsfelder werden intern zunächst in achtstellige CPGWD-Felder konvertiert. Lässt die Feldlänge der Vergleichsfelder keine Werte für Monat und/oder Tag zu, werden diese intern durch '01' ersetzt.

Beispiel: Aus dem zweistelligen Feld, das den Wert 97 enthält, wird für IF-DAT CPGWDx mit dem Wert 19970101.

Beachte: IF-DAT prüft die Vergleichsfelder nicht. Der Programmierer ist dafür verantwortlich, dass die Felder im richtigen Format zur Verfügung stehen.

Unterstützte Formate für IF-DAT

Länge	Alpha	Numerisch mit 0 Dezimalen	intern
2	97	97	19970101
3		097	19970101
		197	19970101

4	1297	1297	19971201
5	12.97	01297	19971201
	12/97	11297	19971201
		21297	20971201
6	311297	311297	19971231
7		0311297	19971231
		1311297	19971231
		0120502	20020512
		1120502	19020512
		2120502	20020512
8	31121997	31121997	19971231
	31.12.97		19971231
	31/12/97		19971231
9		031121997	19971231
10	31.12.1997		19971231
	31/12/1997		19971231

Grundsätzlich arbeitet die interne Routine nach der Regel, dass übergebene Daten nicht verändert werden. Ist ein Vergleichsfeld z.B. 4-stellig alpha mit dem Wert '0097', so wird intern ein Wert '01001997' verarbeitet. Der (nicht vorhandene) Tag wird durch '01' ersetzt, der (falsche) Monat '00' wird nicht verändert.

Bei ungeradstelligen numerischen Feldern steht die erste Stelle für das Jahrtausend: 1 für 1900, 2 für 2000. Für alle anderen Werte wird das 'Fenster' angenommen.

Die IF-DAT-Operationen arbeiten nach der gleitenden Fenstertechnik mit einem Defaultwert von 30. Das heisst, dass Jahreszahlen größer als 30 dem Jahrhundert 19xx zugeordnet werden, Jahre kleiner und gleich 30 dem Jahrhundert 20xx. Ein anderes Fenster kann in der Kundenkonfiguration (CPGURSI2) vorgegeben werden.

IF-DATI Datumsfelder im ISO-Format vergleichen 4370

```
-----------------------------------------------------------
OP F1 OK F2 (BO)
-----------------------------------------------------------
OP   IF-DATI
F1   Name eines Datumsfeldes
OK   Vergleichsoperator (siehe IF)
F2   Name eines Datumsfeldes
BO   logische Verknüpfung mit AND und OR
-----------------------------------------------------------
```

Beispiele:

```
IF-DATI LOEDAT > CPGDAI
IF-DATI ALPHA8 < NUM20

C           LOEDAT    IFGT CPGDAI               I
C           ALPHA8    IFLT NUM20                I
-----------------------------------------------------------
```

Zweck:

Vergleich von Datumsfeldern, die im ISO-Format vorliegen, unabhängig von der Länge. Insbesondere bedeutet das, dass Datumsabfragen mit bis zu sechsstelligen Datumsfeldern auch über den Jahrtausendwechsel hinaus funktionieren.

Beschreibung:

IF-DATI vergleicht intern die Felder CPGWD1 und CPGWD2, die das Datum achtstellig alphanumerisch im ISO-Format enthalten.

IF-DATI kann alphanumerische und numerische Felder miteinander vergleichen, auch wenn Länge und/oder Typ unterschiedlich sind. Die Vergleichsfelder werden intern zunächst in achtstellige CPGWD-Felder konvertiert. Lässt die Feldlänge der Vergleichsfelder keine Werte für Monat und/oder Tag zu, werden diese intern durch '01' ersetzt.

Beispiel: Aus dem vierstelligen Feld, das den Wert 9707 enthält, wird für IF-DATI CPGWDx mit dem Wert 19970701.

Beachte:
- IF-DATI prüft die Vergleichsfelder nicht. Der Programmierer ist dafür verantwortlich, dass die Felder im richtigen Format zur Verfügung stehen.
- Vorsicht bei Vergleichen mit unterschiedlichen Feldlängen !
- Beim Ersetzen funktionierender Vergleiche (mit IF oder COMP) durch IF-DATI-Vergleiche muss man genau untersuchen, ob das Ergebnis gleich sein wird.

Unterstützte Formate für IF-DATI

Länge	Alpha	Numerisch mit 0 Dezimalen	intern
2	97	97	19970101
3		097	19970101
		197	19970101
4	9712	9712	19971201
5	97.12	09712	19971201
	97/12	19712	19971201
6	971231	971231	19971231
7		0971231	19971231
		1971231	19971231
8	19971231	19971231	19971231
	97.12.31		19971231
	97/12/31		19971231
9		019971231	19971231
10	1997.12.31		19971231
	1997/12/31		19971231

Grundsätzlich arbeitet die interne Routine nach der Regel, dass übergebene Daten nicht verändert werden. Ist ein Vergleichsfeld z.B. 4-stellig alpha mit dem Wert '0097', so wird intern ein Wert '01001997' verarbeitet. Der (nicht vorhandene) Tag wird durch '01' ersetzt, der (falsche) Monat '00' wird nicht verändert.

Bei ungeradstelligen numerischen Feldern steht die erste Stelle für das Jahrtausend: 1 für 1900, 2 für 2000. Für alle anderen Werte wird das 'Fenster' angenommen.

Die IF-DAT-Operationen arbeiten nach der gleitenden Fenstertechnik mit einem Defaultwert von 30. Das heisst, dass Jahreszahlen größer als 30 dem Jahrhundert 19xx zugeordnet werden, Jahre kleiner und gleich 30 dem Jahrhundert 20xx. Ein anderes Fenster kann in der Kundenkonfiguration (CPGURSI2) vorgegeben werden.

IF-DATK Datumsfelder mit Kalendertag im ISO-Format verglei- 4373
chen

```
------------------------------------------------------------
OP F1 OK F2 (BO)
------------------------------------------------------------
OP   IF-DATK
F1   Name eines fünf- oder siebenstelligen Datumsfeldes
OK   Vergleichsoperator (siehe IF)
F2   Name eines fünf- oder siebenstelligen Datumsfeldes
BO   logische Verknüpfung mit AND und OR
------------------------------------------------------------
```

Beispiel

```
IF-DATK KTAG1 <= KTAG2

C           KTAG1     IFLE KTAG2                  K
------------------------------------------------------------
```

Zweck:

Vergleich von Kalendertagen, die in der Form jjttt oder jjjjttt in einem fünf- oder siebenstelligen alphanumerischen oder numerischen Feld stehen.

Beachte:

Diese Operation ist in erster Linie eine Umstellungshilfe für den Jahrtausendwechsel. Nach dem Jahr 2000 wird (wie heute) die Operation IF zur Abfrage ausreichen.

Intern werden die Daten in der Form jjjjttt0 bereitgestellt und verglichen.

Die Operation steht im QPG nicht zur Verfügung.

INDOF Bezugszahlen löschen 4375

```
------------------------------------------------------------
(ON) OP (BD)
------------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   'INDOF' muss eingetragen werden
BD   zwei Bezugszahlen zur Beschreibung eines Bereichs
------------------------------------------------------------
```

Beispiele:

```
- INDOF                    * alle Schalter 01 - 99
- INDOF  21 39             * nur Schalter  21 - 39
------------------------------------------------------------
```

Zweck:

Ein Bezugszahlenbereich soll gelöscht werden.

Beschreibung:

Die Operation INDOF löscht einen Bereich von Bezugszahlen. Wird in BD kein Schalter angegeben, so werden alle Bezugszahlen von 01 bis 99 gelöscht. C-, P- und T-Schalter bleiben unverändert.

Der zu löschende Bereich kann durch Angabe zweier aufsteigend geordneter Bezugszahlen eingeschränkt werden.

INDON Bezugszahlen setzen 4380

Zweck:

Ein Bezugszahlenbereich soll gesetzt werden.

Beschreibung:

Die Operation INDON arbeitet analog zu INDOF.

JLB Linksbündig verschieben, Blanks nach hinten 4382

```
-------------------------------------------------------
(ON) OP F2
-------------------------------------------------------

On   Bedingungsabfrage (bis zu drei Schalter)
OP   JLB muss eingetragen werden
F2   Name eines alphanumerischen Feldes
-------------------------------------------------------
```

Zweck:

Linksbündiges Verschieben eines Feldinhalts.

Beschreibung:

Mit dieser Operation kann der Inhalt eines Alphafeldes so verschoben werden, dass das erste Zeichen in der ersten Stelle des Feldes steht.

Von rechts wird das Feld mit Blanks aufgefüllt.

Beispiel: - JLB FELD

```
Feldinhalt vorher  1.   (   123)   2.   (  12 3)
Feldinhalt nachher      (123   )        (12 3  )
```

JRB Rechtsbündig verschieben, Blanks nach vorne 4385

```
-------------------------------------------------------
(ON) OP F2
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   JRB oder RIGHT muss eingetragen werden
F2   Name eines  alphanumerischen Feldes
-------------------------------------------------------
```

Beispiel :

```
- JRB  FELD
- RIGHT  FELD
-------------------------------------------------------
```

Zweck:

Rechtsbündiges Verschieben eines Feldinhalts.

Beschreibung:

Mit dieser Operation kann der Inhalt eines Alphafelds so verschoben werden, dass das letzte Zeichen ungleich Blank nach der Ausführung in der rechtesten Stelle des Feldes steht.

Von links wird das Feld mit Blanks aufgefüllt.

Beispiele: - JRB FELD

Feldinhalt vorher 1. (123) 2. (12 3)
Feldinhalt nachher (123) (12 3)

Anwendung: Randausgleich für Textausgaben,
ungepackte numerische Daten, usw.

Beachte: Beispiel 2 erläutert die genaue Arbeitsweise von JRB: Es werden nicht lediglich die Blanks nach links sortiert; Zeichenketten bleiben unverändert, auch wenn sie Blanks einschließen. Um in diesem Fall das gleiche Ergebnis wie in Beispiel 1 zu erzielen, müsste vor dem JRB ein

- DELC ' ' FELD

programmiert werden.

JRC Rechtsbündig verschieben, best. Zeichen nach vorne 4390

(ON) OP F2 EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP JRC oder RIGHT-CHAR muss eingetragen werden
F2 Zeichen, mit dem aufgefüllt wird, in Hochkommata
EG Name eines alphanumerischen Feldes

Beispiel :

- JRC '*' FELD
- RIGHT-CHAR '-' FELD2

Zweck:

Rechtsbündiges Verschieben eines Feldinhalts.

Beschreibung:

Die Operation JRC arbeitet wie JRB (s.o.). Die nach links sortierten Blanks werden mit dem Zeichen überschrieben, das in F2 angegeben ist.

Beispiel: - JRC '*' FELD

Feldinhalt vorher (123)
Feldinhalt nachher (***123)

JRZ Rechtsbündig verschieben, Nullen nach vorne 4395

```
-----------------------------------------------------
(ON) OP EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   JRZ  oder  RIGHT-ZERO  muss eingetragen werden
EG   Name eines alphanumerischen Feldes
-----------------------------------------------------

Beispiel :

- JRZ   FELD
- RIGHT-ZERO  KUNDENNUMMER
-----------------------------------------------------

Zweck:

Rechtsbündiges Verschieben eines Feldinhalts.

Beschreibung:

Die Operation  JRZ  arbeitet wie JRB ( s.o. ).  Die nach
links sortierten Blanks werden mit Nullen überschrieben

Beispiel:             - JRZ FELD

Feldinhalt vorher          (123   )
Feldinhalt nachher         (000123)
```

LEFT-SHIFT Linksbündig verschieben, Blanks nach hinten 4397

```
-----------------------------------------------------
(ON) OP EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   LEFT-SHIFT muss eingetragen werden
EG   Names eines alphanumerischen Feldes
-----------------------------------------------------

Beispiel:

- LEFT-SHIFT   FELD
-----------------------------------------------------

Zweck:

Linksbündiges Verschieben eines Feldinhalts.

Beschreibung:

Die Operation LEFT-SHIFT arbeitet wie die Operation JLB.
```

LIST Ausgabe extern beschriebener Listen 4400

```
-----------------------------------------------------
(ON) (F1) OP F2 (DY) (SV) (EG)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Druckername ( nur online erforderlich )
OP   LIST  muss eingetragen werden
F2   Name eines QTF-Dokuments oder 32-stelligen Feldes
DY   SECTION kann zur Dokumentation codiert werden
EG   Name einer Section im Druckdokument
SV   VARIABEL oder VARIABLE, TYPE P oder TYPE I
-----------------------------------------------------
```

Beispiele:

```
- 'L86C' LIST KUNDE
-  DRID  LIST DOKUM5  SECTION  KOPF
-        LIST  VARLST  VARIABEL
-  DRID  LIST  DOKUM  SECTION ENDE  TYPE PHASE
-----------------------------------------------------
```

Zweck:

Drucken einer Liste, die programmextern im QTF (Quick Text Facility) beschrieben ist.

Beschreibung:

In F1 muss (außer bei Batch-Programmierung und variabler Verarbeitung) der Druckername als Konstante oder als Feld angegeben werden.

In F2 wird der Name des QTF-Dokuments angegeben, das die Beschreibung der Liste enthält. Dieses Dokument muss im QTF in der Library LIST abgestellt sein.

Zusätzlich kann der Name einer Section angegeben werden; eine Section ist ein Teil eines Dokuments.

Wird die definierte Seitenlänge überschritten, so wird der Schalter OF (Overflow) gesetzt.

LIST kann auch variabel programmiert werden. In diesem Fall wird nur der Opcode, der Name eines 32-stelligen Feldes, das alle benötigten Informationen für den Ausdruck enthält und das Schlüsselwort VARIABLE oder VARIABEL angegeben.

Für das variable LIST empfehlen wir den separaten Befehl LIST-VAR (s.u.)

Mit dem Service TYPE kann auf LIST-Phasen zugegriffen werden. TYPE P: Es wird nicht auf ein Dokument zugegriffen, sondern auf eine Phase. TYPE I: Es wird auf eine Phase zugegriffen, wenn kein Dokument vorhanden ist.

Ausführliche Beispiele finden Sie im QTF-Handbuch.

LIST-VAR Variable Programmierung des LIST-Befehls 4401

(ON) OP EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP LIST-VAR muss eingetragen werden
EG Name eines 32-stelligen Alphafeldes

Zweck:

Der LIST-Befehl (wie oben beschrieben) soll variabel programmiert werden. Die notwendigen Daten werden zur Ausführungszeit dem 32-stelligen Feld entnommen.

Aufbau des 32-stelligen Feldes (alle Teilfelder alpha):

Stelle 1 - 8	Dokument
Stelle 9 - 14	Section
Stelle 15 - 18	Drucker-Id
Stelle 19 - 22	Library
Stelle 23 - 23	Drucker Exit (siehe QTF-Handbuch)
Stelle 24 - 24	Übersetzen (N, 1 oder 2)
Stelle 25 - 25	Phasen-Verarbeitung (P oder I)
Stelle 26 - 32	noch nicht belegt

LOADT Geretteten Bildschirminhalt zurückladen 4405

```
-------------------------------------------------------
(ON) OP F2 (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   LOADT  muss eingetragen werden
F2   Name einer Temporary Storage Queue
SV   VARiabel oder Save
-------------------------------------------------------
```

Beispiele:

- LOADT TS01
- LOADT XX VAR

Zweck:

Laden eines mit SAVET geretteten Bildschirminhalts.

Beschreibung:

Diese Operation schreibt den mit SAVET (s.u.) geretteten Bildschirminhalt wieder zurück auf das Terminal. Es muss der Name der Temporary Storage Queue angegeben werden,auf der mit der Operation SAVET die Daten abgestellt wurden.

Der Name kann konstant vierstellig angegeben werden oder in einem Feld, das vierstellig alphanumerisch definiert ist. Im zweiten Fall gibt man zusätzlich VARiabel an.

Die Operation LOADT ist nur zusammen mit SAVET sinnvoll; dabei ist es aber nicht erforderlich, dass die beiden Operationen im gleichen Programm verwendet werden.

Die Servicefunktion Save ermöglicht es, einen geretteten Bildschirm mehrfach zu laden. Der Zwischenspeicherbereich bleibt so lange erhalten, bis LOADT ohne Service S ausgeführt wird (bzw. bis zum CICS Shut Down).

Bei einem Fehler wird von CPG der Schalter EF gesetzt.

Bei dialogorientierter Programmierweise hält LOADT das Programm nicht an. Dazu muss ein Bild noch explizit eingelesen werden.

Beachte:

Das Zurückladen der Farben ist nur dann gewährleistet, wenn sie per Farbattribut (B,G,P,R,T,W,Y) gesetzt werden

LOADT-VAR Geretteten Bildschirminhalt zurückladen 4407

```
-----------------------------------------------------
(ON) OP F2 (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   LOADT-VAR  muss eingetragen werden
F2   Name eines vierstelligen Alphafelds für TS-Namen
SV   Save
-----------------------------------------------------
```

Beispiele:

```
- LOADT-VAR  FELD
-----------------------------------------------------
```

Zweck:

Laden eines mit SAVET geretteten Bildschirminhalts (siehe LOADT mit Servicefunktion VAR).

LOKUP Suchen in einer Feldgruppe 4410

```
-----------------------------------------------------
OP F1 OK F2
-----------------------------------------------------

OP   LOKUP oder LOOK-UP muss eingetragen werden
F1   Name der zu durchsuchenden Feldgruppe
OK   Operator
F2   Suchargument
-----------------------------------------------------
```

Beispiele:

```
-  LOKUP FG(I) = FELD
-  LOKUP FG(I) GE FELD
-----------------------------------------------------
```

Zweck:

Eine Feldgruppe soll auf einen bestimmten Inhalt untersucht werden.

Beschreibung:

Dieser Befehl ermöglicht es, Feldgruppen auf einen bestimmten Inhalt hin zu untersuchen. Der zu suchende Inhalt wird entweder als alphanumerisches Literal oder als variables Feld angegeben.

Die angegebene Feldgruppe wird elementweise mit dem Suchargument auf Identität verglichen. Sobald die Bedingung erfüllt ist, wird die Operation LOKUP beendet.

Wird die Operation LOKUP ohne Operator verwendet, so wird auf 'gleich' gesucht.

Bei der Operation LOKUP muss ein Index angegeben werden, welcher dann folgende Funktion erfüllt:

1. Der Vergleich beginnt erst bei dem durch den Inhalt des Indexfeldes gekennzeichneten Feldgruppenelement.

2. Nach Beendigung der LOKUP-Operation steht im Indexfeld der Index des Elementes, das die Bedingung des Vergleiches erfüllt hat bzw. 0 (Null), wenn die Bedingung von keinem Element der Feldgruppe erfüllt wurde.

Beispiel : Die 5-elementige Feldgruppe FG3 habe folgenden Inhalt :

```
CPG
HL1
QFF
QSF
QTF
```

Der Befehl
- LOKUP FG3(I) = 'HL1'

hat folgende Auswirkungen :

Ist vor dem LOKUP z.B. I=1, dann wird 'HL1' gefunden; damit ist nach dem LOKUP I=2.

Ist vor dem LOKUP z.B. I=3, dann werden nur die Elemente 3 bis 5 mit dem Argument 'HL1' verglichen und I=0 gesetzt.

Zu beachten ist, dass das Suchargument und die Feldgruppe in der Länge übereinstimmen müssen. Im obigen Beispiel wäre etwa ein Suchargument 'Q' wegen der Länge 1 nicht zulässig.

LOOK-UP Suchen in einer Feldgruppe 4415

identisch mit LOKUP (s.o.)

MACRO Assembler-Instruktion einfügen 4420

```
-----------------------------------------------------
(ON) (F1) OP F2
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Label
OP   MACRO  muss eingetragen werden
F2   Assembler-Instruktion ( bis zu 42 Stellen )
-----------------------------------------------------

Beispiel :

-        MACRO  MVC    AUSG(5),EING
-----------------------------------------------------
```

Zweck:

Eine Assembler-Instruktion (auch Macro-Instruktion) soll ins Programm eingefügt werden.

MAP 1. Map lesen 4425
2. Map-Felder im Programm löschen
3. Variable Attribute einer Map löschen

---

1. (ON) OP F2 (SV1) (SV2)
2/3. (ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MAP oder RECEIVE muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV1 Service: 'VAR' für variablen Mapnamen
SV2 Service: 'LOW' für Kleinbuchstaben
SV Service: B löscht Felder, C löscht Attribute

Beispiele:

- MAP KUNDEN.
- RECEIVE BRIEF LOW.
- MAP VAR16 VAR LOW

Zweck:

1. Aus einer QSF-Map sollen Daten gelesen werden.
2. Alle Mapfelder sollen im Programm gelöscht werden
3. Alle variablen Attributfelder einer Map sollen im im Programm gelöscht werden.

Beschreibung zu 1. :

Dieser Befehl ermöglicht es, alle Bildschirmfelder interaktiv außerhalb des Programms zu beschreiben.

Anwendungsprogramme definieren keine Bildschirm-Eingaben und -Ausgaben. Alle Konstanten und variablen Felder werden programmextern beschrieben. Eine Veränderung der Bildschirmmaske erfordert in der Regel keine erneute Umwandlung des Anwendungsprogramms. Alle im Programm definierten Felder können in eine Map aufgenommen werden.

Die Operation MAP liest transaktionsorientiert Daten vom Bildschirm ein. Dieser Befehl kann am Programmanfang gegeben werden. In F2 wird der Name der Map eingetragen, der die Felder der Terminal I-O-Area zugeordnet werden. Enthält SV das Schlüsselwort 'LOW', so wird vom QSF keine Übersetzung in Großbuchstaben durchgeführt. Dies setzt allerdings voraus, dass in der Terminal-Con-Trol-Tabelle des TP-Systems UCTRAN nicht gesetzt ist, bzw. durch die Operation UCTRAN OFF im Programm ausgeschaltet wird.

Das Schlüsselwort 'VAR' ermöglicht die Verwendung eines variablen Mapnamens. Hierzu ist es erforderlich, den Namen der zu bearbeitenden Map per Programm in ein 16-stelliges alphanumerisches Feld zu übertragen. Der Aufbau des 16-stelligen Feldes ist weiter unten beschrieben

Beschreibung zu 2. :

Der Befehl MAP MASKE1 Blank löscht im Programm alle Felder, die in der Map vorkommen, auch die Felder der zusätzlichen Eingabe. Ein Map-Input oder -Output findet dabei nicht statt. Der Service B kann alternativ auch bei variabler Mapverarbeitung in Stelle 12 des 16-stelligen Kontrollfeldes gesetzt werden. Alphafelder werden auf Blank gelöscht, numerische Felder auf Null.

Beschreibung zu 3. :

Der Befehl MAP MASKE2 Clear löscht im Programm alle variablen Attributfelder, die in der Map angegeben sind. Ein Map-Input oder Map-Output findet dabei nicht statt. Der Service C kann alternativ auch bei variabler Mapverarbeitung in Stelle 12 des 16-stelligen Kontrollfeldes gesetzt werden.

Das 16-stellige Kontrollfeld für variable Mapverarbeitung hat folgenden Aufbau:

1-8 Mapname

9 'Y' bedeutet, dass der Bildschirm vorher gelöscht wird.

'N' bedeutet, dass kein Erase durchgeführt wird.

'Y' und 'N' haben Vorrang vor dem im QSF definierten Erase-Eintrag.

' ' der Erase-Eintrag aus der QSF-Definition wird verwendet.

10 Write Control Character.Siehe Kapitel Tabellen 7030

11 'F' bedeutet, nur variable Felder ausgeben. Dies bewirkt eine Minimierung der Datenübertragung bei entfernt betriebenen Bildschirmen.
'S' bedeutet, nur variable Felder ausgeben, und zwar ohne Attribut

12 ' ' Funktion MAP (transaktionsorientierte Eingabeübertragung wird ausgeführt.)
'B' Löschen der Mapfelder auf Blank oder Null im Programm (s.o.)
'C' Löschen der variablen Attributfelder der Map im Programm.
'I' Funktion MAPI (dialogorientierter Map-Input wird ausgeführt.)
'O' Funktion MAPO. Die Maske wird ausgegeben.

Alle anderen Einträge werden in 'O' umgesetzt.

13-16 reserviert.

MAP-VAR MAP-Befehl variabel 4428

```
-------------------------------------------------------
(ON) OP EG (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   MAP-VAR  muss eingetragen werden
EG   16-stelliges Feld mit variablen Informationen
SV   Service: 'LOW' für Kleinbuchstaben
-------------------------------------------------------
```

Zweck:

Der MAP-Befehl (s.o.) soll variabel programmiert werden. Der Aufbau des 16-stelligen Alphafeldes, dem zur Ausführungszeit die notwendigen Daten entnommen werden, ist beim Befehl MAP beschrieben

MAPD Map-Dialog 4430

```
-------------------------------------------------------
(ON) OP F2 (SV1) (SV2)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   MAPD  muss eingetragen werden
F2   enthält den bis zu 8-stelligen Namen der Map
SV1  VAR  für variablen Mapnamen
SV2  Service: LOW, CLEar, A, C, K, S, T
-------------------------------------------------------
```

Beispiele:

```
- MAPD KUNDEN.
- ON P1 MAPD NUMMER VAR S
```

```
-------------------------------------------------------
```

Zweck:

Eine QSF-Map soll ausgegeben werden, und anschließend sollen Daten aus dieser Map gelesen werden.

Beschreibung:

Dieser Befehl ermöglicht es, alle Bildschirmfelder interaktiv außerhalb des Programms zu beschreiben.

Anwendungsprogramme definieren keine Bildschirm-Eingaben und -Ausgaben. Alle Konstanten und variablen Felder werden programmextern beschrieben. Eine Veränderung der Bildschirmmaske erfordert in der Regel keine erneute Umwandlung des Anwendungsprogramms. Alle im Programm definierten Felder können in eine Map aufgenommen werden.

Die Operation MAPD ist eine Kombination der Operationen MAPO und MAPI. In F2 wird ein Mapname eingetragen, der sowohl für die Ein- als auch für die Ausgabe verwendet wird.

Das Schlüsselwort 'VAR' ermöglicht die Verwendung eines variablen Mapnamens im Programm. Die Vorgehensweise ist beim Befehl MAP ausführlich beschrieben (s.o.).

Enthält SV2 das Schlüsselwort 'LOW', so wird vom QSF keine Übersetzung in Großbuchstaben durchgeführt.

Das Schlüsselwort 'CLEar' ermöglicht die Abfrage des Schalters CL (Löschtaste). Anderenfalls bewirkt diese Taste die Beendigung des Programms. Hardwarebedingt wird mit der Löschtaste immer der Bildschirm gelöscht.

Service 'S' steht für die Kombination von CLEAR und LOW

Für 'quasi-transaktionsorientiertes' Programmieren stehen weitere Services zur Verfügung, die bei der Operation MAPI (nächste Seite) beschrieben sind.

MAPD-VAR Variabler Map-Dialog 4431

```
-------------------------------------------------------
(ON) OP EG (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   MAPD-VAR  muss eingetragen werden
EG   16-stelliges Feld mit variablen Informationen
SV   Services: LOW, CLEar, A, C, K, S, T
-------------------------------------------------------
```

Zweck:

Der MAPD-Befehl (s.o) soll variabel programmiert werden. Der Aufbau des 16-stelligen Alphafeldes, dem zur Ausführungszeit die notwendigen Befehle entnommen werden, ist beim Befehl MAP beschrieben.

MAPI Eingabe QSF-Map 4435

(ON) OP F2 (SV1) (SV2)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MAPI muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV1 Service: VAR für variablen Mapnamen
SV2 Service: LOW, CLEar, A, C, K, S, T

Beispiele:

- MAPI ARTIKEL.
- ON P2 AND NOT 10 MAPI BRIEF LOW.

Zweck:

Aus einer Map sollen Daten im Dialog gelesen werden.

Beschreibung:

Dieser Befehl ermöglicht es, alle Bildschirmfelder interaktiv außerhalb des Programms zu beschreiben.

Anwendungsprogramme definieren keine Bildschirm-Eingaben und -Ausgaben. Alle Konstanten und variablen Felder werden programmextern beschrieben. Eine Veränderung der Bildschirmmaske erfordert in der Regel keine erneute Umwandlung des Anwendungsprogramms. Alle im Programm definierten Felder können in eine Map aufgenommen werden.

Die Operation MAPI liest Daten im Dialog vom Bildschirm ein. Dieser Befehl kann mehrmals im Programm gegeben werden. In F2 wird der Name der Map eingetragen, aus der die Felder gelesen werden.

Das Schlüsselwort 'VAR' ermöglicht die Verwendung eines variablen Mapnamens im Programm. Die Vorgehensweise ist beim Befehl MAP ausführlich beschrieben (s.o.).

Enthält SV das Schlüsselwort 'LOW', so wird vom QSF keine Übersetzung in Großbuchstaben durchgeführt.

Das Schlüsselwort 'CLEar' ermöglicht die Abfrage des Schalters CL (Löschtaste). Anderenfalls bewirkt diese Taste die Beendigung des Programms. Hardwarebedingt wird mit der Löschtaste immer der Bildschirm gelöscht.

Service 'S' steht für die Kombination von CLEAR und LOW

Für 'quasi-transaktionsorientiertes' Programmieren stehen vier weitere Services zur Verfügung:

T für die transaktionsorientierte Programmierweise
A impliziert T und LOW
C impliziert T und CLEar
K impliziert T, LOW und CLEar

Achtung: Groß-/Kleinschreibung ist beim transaktionsorientierten Programmieren nur möglich, wenn in der TCT der Parameter UCTRAN nicht definiert wurde. Außerdem muss in der OPTIONS-Karte 'LOW' eingetragen sein.

Hinweis: Diese 'quasi-transaktionsorientierte' Programmierweise unterliegt einigen Einschränkungen.

Sie darf nicht eingesetzt werden
- in Programmen, die mit EXPR oder EXITP mit Programmnamen aufgerufen werden
- in HL1 - Bausteinen
- wenn ein Dataset im Programm definiert ist
- wenn ein CHAIN für Update über der Operation liegt (CHAIN U wird dann ignoriert).

Sie sollte nur nach ausreichender Schulung angewandt werden.

MAPI-VAR MAPI-Befehl variabel 4436

(ON) OP EG (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MAPI-VAR muss eingetragen werden
EG 16-stelliges Feld mit variablen Informationen
SV Services: LOW, CLEar, A, C, K, S, T

Zweck:

Der MAPI-Befehl (s.o) soll variabel programmiert werden. Der Aufbau des 16-stelligen Alphafeldes, dem zur Ausführungszeit die notwendigen Befehle entnommen werden, ist beim Befehl MAP beschrieben.

MAPO QSF-Map ausgeben 4440

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MAPO oder SEND muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV VAR für variablen Mapnamen

Beispiele:

- MAPO MENUE.
- ON EF SEND FEHLER.
- MAPO MAPVAR VAR

Zweck:

Eine QSF-Map soll ausgegeben werden.

Beschreibung:

Dieser Befehl ermöglicht es, alle Bildschirmfelder interaktiv außerhalb des Programms zu beschreiben.

Anwendungsprogramme definieren keine Bildschirm-Eingaben und -Ausgaben. Alle Konstanten und variablen Felder werden programmextern beschrieben. Eine Veränderung der Bildschirmmaske erfordert in der Regel keine erneute Umwandlung des Anwendungsprogramms. Alle im Programm definierten Felder können in eine Map aufgenommen werden.

Die Operation MAPO schreibt die in F2 angegebene Map auf den Bildschirm.

Die Operation MAPO gibt eine separat erstellte QSF-Map aus. Dieser Befehl kann mehrmals im Programm gegeben werden. In F2 wird der Name der Map eingetragen, die auf den Bildschirm ausgegeben werden soll.

Das Schlüsselwort 'VAR' ermöglicht die Verwendung eines variablen Mapnamens im Programm. Die Vorgehensweise ist beim Befehl MAP ausführlich beschrieben (s.o.).

MAPO-VAR Variabele Map-Ausgabe 4441

```
-------------------------------------------------------
(ON) OP EG
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   MAPO-VAR  muss eingetragen werden
EG   16-stelliges Feld mit variablen Informationen
-------------------------------------------------------
```

Zweck:

Der MAPO-Befehl (s.o) soll variabel programmiert werden. Der Aufbau des 16-stelligen Alphafeldes, dem zur Ausführungszeit die notwendigen Befehle entnommen werden, ist beim Befehl MAP beschrieben.

MAPP QSF-Map auf einen Drucker ausgeben 4445

```
-------------------------------------------------------
(ON) F1 OP F2 (SV1) (SV2)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Name des Online-Druckers ( als Feld oder konstant )
OP   MAPP  muss eingetragen werden
F2   enthält den bis zu 8-stelligen Namen der Map
SV1  VAR  für variablen Mapnamen
SV2  AFTer, BEFore, S  für Vorschub auf den Blattanfang
-------------------------------------------------------
```

Beispiele:

- DRID MAPP ARTIKEL
- 'DR15' MAPP OPOSTEN BEFORE

Zweck:

Eine QSF-Map soll auf einen Online-Drucker ausgegeben werden.

Beschreibung:

Die Operation MAPP druckt die in F2 eingetragene Map auf den in F1 definierten Online-Drucker. F1 kann ein variables 4-stelliges Alphafeld oder auch eine Konstante sein.

Achtung:

Es werden nur Zeilen gedruckt, in denen sich zumindest ein Zeichen befindet. Sollen also 24 Zeilen gedruckt werden, so muss in jeder Zeile der Map zumindest ein

Blank (dargestellt durch ein '#') beschrieben werden.

Das Schlüsselwort 'VAR' ermöglicht die Verwendung eines variablen Mapnamens im Programm. Die Vorgehensweise ist beim Befehl MAP ausführlich beschrieben (s.o.).

Folgende weitere Services sind unterstützt:

AFT - Nach dem MAPP wird ein Vorschub auf Blattanfang durchgeführt.
BEF - Vor dem MAPP wird ein Vorschub auf Blattanfang durchgeführt.
'S' - Vor und nach dem MAPP wird ein Vorschub auf den Blattanfang durchgeführt.

MAPP-VAR MAPP-Befehl variabel 4446

```
-----------------------------------------------------
(ON) F1 OP EG (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Online-Drucker als Feld  oder als Konstante
OP   MAPP-VAR  muss eingetragen werden
EG   16-stelliges Feld mit variablen Informationen
SV   AFTer, BEFore, S  für Vorschub auf den Blattanfang
-----------------------------------------------------
```

Zweck:

Der MAPP-Befehl (s.o) soll variabel programmiert werden. Der Aufbau des 16-stelligen Alphafeldes, dem zur Ausführungszeit die notwendigen Operationen entnommen werden, ist oben beim Befehl MAP beschrieben.

MLLZO Move low to low Zone 4450

(ON) OP F2 EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP MLLZO muss eingetragen werden
F2 Feld, dessen Zone übertragen wird
EG Feldname des Ergebnisfeldes

Beispiele:

- MLLZO 1 ZAHL
- MLLZO '1' ALPHA
- MLLZO BIT8 KEY

Zweck:

Übertragen der Zone des rechtesten Bytes des in F2 angegebenen Feldes auf das rechteste Byte des Ergebnisfeldes.

Beschreibung :

Die Operation MLLZO ist für alphanumerische und numerische Felder unterstützt.

Bei numerischen Feldern sind nur Ziffern als F2 zugelassen. Es wird also grundsätzlich das Zonenhalbbyte 'F' in das Feld in EG übertragen.

Für alphanumerische Felder kann F2 einen Feldnamen oder eine Konstante enthalten.

Beispiel : MLLZO '1' FELD

Feldinhalt vorher (F4 F3 F2 C1) (432A)
Feldinhalt nachher (F4 F3 F2 F1) (4321)

MOVE(R) Rechtsbündig übertragen 4455

(ON) OP F2 (DY) EG (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MOVE, MOVER oder MOVE-RIGHT ist einzutragen
F2 Feldname des Ursprungsfeldes
EG Feldname des Ergebnisfeldes
SV Service: 'C', 'F', 'I', 'INDex'
DY Dummywort: 'TO'

Beispiele:

```
- MOVE A B.
- ON EF MOVER A TO B.
- ON PF1 MOVE-RIGHT 'X' TO TEST.
- MOVE 3,14 TO PI.
- MOVE UMS(M) TO UMSATZ.
- MOVE FG(1) TO FG(I)
- MOVE '*' TO STERN.
- MOVE X'EFEFEF' TO ENDTAB.  * Hexadezimale Konstante,
                             * (bis zu drei Bytes)
```

Zweck:

Der Inhalt eines Feldes soll in ein anderes Feld übertragen werden.

Beschreibung:

F2 wird von rechts nach links nach EG übertragen.

Die Felder können alphanumerisch, numerisch oder unterschiedlich definiert sein.

Diese Operation ist voll indizierbar.

Bei einer MOVE-Operation numerisch nach alpha kann die rechte Zone des Alpha-Feldes verändert werden. Dies geschieht über eine Eintragung in SV. Ein 'C' oder 'F' bedeutet, dass abweichend von der bei der Installation festgelegten Standardannahme das Vorzeichen 'C' oder 'F' gelten soll.

Ein 'I' oder 'INDex' bewirkt, dass keine Prüfung eines benutzten Index durchgeführt wird. Der Programmierer trägt dann die Verantwortung dafür, dass kein Speicherbereich irrtümlich überschrieben wird.

MOVEA Bereich linksbündig übertragen 4460

(ON) OP F2 (DY) EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP MOVEA oder MOVE-ARRAY muss eingetragen werden
F2 zu übertragender Bereich
DY Dummywort 'TO'
EG Feldname des Ergebnisfeldes

Beispiele:

- MOVEA F256 TO FG16
- MOVEA FG(IND) LINE24
- MOVE-ARRAY FG1 TO FG2

Zweck:

Übertragen alphanumerischer Zeichen, wobei entweder der zu übertragende Bereich oder das Ergebnisfeld eine Feldgruppe ist bzw. beide Feldgruppen sind.

Beschreibung :

Die Operation MOVEA überträgt alphanumerische Zeichen beginnend mit der linken Stelle von F2 nach EG; im Gegensatz zu den Operationen MOVE und MOVEL werden Feldgruppen nicht elementweise verschoben, sondern als ein zusammenhängender Bereich.

Die Länge der MOVEA-Operation wird durch die Länge des kürzeren Feldes (F2 oder EG) bestimmt.

Beispiele :

```
Feldinhalte vorher :  FG25 ( ABCDE )   FG34  ( **** )
                           ( FGHIJ )         ( **** )
                                             ( **** )

- MOVEA  FG25 TO FG34                        ( ABCD )
                                             ( EFGH )
                                             ( IJ** )

- MOVE-ARRAY  FG25(2) TO  FG34(2)            ( **** )
                                             ( FGHI )
                                             ( J*** )
```

MOVEL Linksbündig übertragen 4465

(ON) OP F2 (DY) EG (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MOVEL oder MOVE-LEFT muss eingetragen werden
F2 Feldname des Ursprungsfeldes
EG Feldname des Ergebnisfeldes
SV Service: 'C', 'F', 'I', 'INDex'
DY Dummywort: 'TO'

Beispiele:

```
- MOVEL A B.
- ON EF MOVE-LEFT  A  TO  B
- ON PF1 MOVEL 'X' TO TEST.
- MOVEL UMS(M) TO UMSATZ.
- MOVEL FG(1) TO FG(I).
- MOVE-LEFT  STERN  TO  HEINRICH.
- MOVEL X'0D0C0D' TO ESCAPE.    * Hexadezimale Konstante
                                * (bis zu drei Bytes)
```

Zweck:

Der Inhalt eines Feldes soll in ein anderes Feld übertragen werden.

Beschreibung:

F2 wird von links nach rechts nach EG übertragen.

Die Felder können alphanumerisch, numerisch oder unterschiedlich definiert sein.

Diese Operation ist voll indizierbar.

Bei einer MOVEL-Operation numerisch nach alpha kann die rechte Zone des Alpha-Feldes verändert werden. Dies geschieht über eine Eintragung in SV. Ein 'C' oder 'F' bedeutet, dass abweichend von der bei der Installation festgelegten Standardannahme das Vorzeichen 'C' oder 'F' gelten soll.

Ein 'I' oder 'INDex' bewirkt, dass keine Prüfung eines benutzten Index durchgeführt wird. Der Programmierer trägt dann die Verantwortung dafür, dass kein Speicherbereich irrtümlich überschrieben wird.

MOVEN Alphanumerisches in numerisches Feld übertragen 4470

```
-------------------------------------------------------
(ON) OP F2 (DY) EG (SV) (BD)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   MOVEN  muss eingetragen werden
F2   Feldname des alphanumerischen Ursprungsfeldes
EG   Feldname des numerischen Ergebnisfeldes
SV   'H' für erweiterte Prüfung ungültiger Zeichen
BD   bis zu drei Schalter können angegeben werden
DY   Dummywort: 'TO'
-------------------------------------------------------
```

Beispiele:

```
- MOVEN A N
- ON NI MOVEN  A  TO  B
- MOVEN UMS(M) TO UMSATZ  H
- MOVEN FGA(1) TO FGN(I)  10 20 30
-------------------------------------------------------
```

Zweck:

Der Inhalt eines alphanumerischen Feldes soll in ein numerisches Feld übertragen werden, und zwar so, dass es der Bildschirmeingabe entspricht.

Beschreibung:

Das Feld in Faktor 2 wird so in das Ergebnisfeld übertragen, wie das Feld vom Bildschirm in ein numerisches Feld eingelesen würde.

Das bedeutet, dass eine Dezimalstellenanpassung vorgenommen wird und gegebenenfalls die Stellen mit dem höchsten Wert vorne abgeschnitten werden. Das Komma wird als Separator der Dezimalstellen erkannt. Minuszeichen und die Zeichenfolge 'CR' im Alphafeld bewirken, dass das Feld als negativer Wert interpretiert wird. Ungültige Zeichen werden aus dem Feld eliminiert und das Feld wird vor dem MOVEN entsprechend komprimiert.

Es können drei Bezugszahlen angegeben werden:

Die erste wird gesetzt, wenn das Alphafeld ungültige Zeichen enthält.

Die zweite wird gesetzt, wenn das Alphafeld vor oder nach einem Komma zu viele Stellen enthält.

Die dritte wird gesetzt, wenn das Alphafeld Blank ist.

Beispiele: MOVEN ALPHA TO NUM 11 12 13

Das numerische Feld sei siebenstellig mit 2 Dezimalstellen definiert.

ALPHA	NUM	intern	gesetzte Bezugszahl
123	123,00	0012300C	
123	123,00	0012300C	
1 2 3	123,00	0012300C	
-123	123,00-	0012300D	
12-3	123,00-	0012300D	
987,65CR	987,65-	0098765D	
,1	0,10	0000010C	
12.345,678	12345,67	1234567C	12, zu viele Dezim.
56,7D	56,70	0005670C	11, ung. Zeichen
ELF DM	0,00	0000000F	11, ung. Zeichen
	0,00	0000000F	13, Feld ist blank
12A4567	24567,00	2456700C	11 und 12

Mit der Servicefunktion H wird eine erweiterte Prüfung ungültiger Zeichen erreicht. Der Schalter für ungültige Zeichen wird bei H auch dann gesetzt, wenn innerhalb von Ziffernfolgen Blanks oder Minuszeichen auftreten:

1 2 3	123,00	0012300C	11, ung. Zeichen(H)
12-3	123,00-	0012300D	11, ung. Zeichen(H)

Grundsätzlich gilt bei der Fehlerprüfung: Sobald der Schalter für die falsche Anzahl von Dezimalstellen gesetzt ist (Schalter 2) wird die Prüfung abgebrochen. Geprüft wird byteweise von links nach rechts.

Beispiel: (Sonderfall)

In ein 7-stelliges Feld ohne Dezimalstellen wird mit dem Service H die Zeichenkette ' 1-,5- ' geschoben. Es müsste der erste Schalter für 'ungültiges Zeichen' gesetzt werden (Minuszeichen mitten im Feld). Da aber nach dem Komma die 5 als falsche Dezimalstelle erkannt und somit der entsprechende Schalter 2 gesetzt wird, ist die bisher geprüfte Zeichenkette nicht fehlerhaft. Schalter 1 bleibt deshalb aus.

MOVEV variable MOVE-Operation 4475

(ON) OP F2 (DY) EG (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP MOVEV muss eingetragen werden
F2 10-stelliges alphanumerisches Feld
EG 10-stelliges alphanumerisches Feld
SV ARRay, LEFt, Numeric, RIGht
DY Dummywort: 'TO'

Beispiele:

- MOVEV A B.
- MOVEV A TO B LEFT
- MOVEV F1 TO F2 ARRAY

Zweck:

Der Inhalt des Feldes, dessen Name in F2 steht, soll in das Feld, dessen Name in EG steht, übertragen werden.

Beschreibung:

Diese Operation ermöglicht es, den Programmablauf von außen, z.B. über eine Tabelle, zu beeinflussen.

Bei gleicher Feldlänge und gleichem Feldtyp werden alphanumerische Felder linksbündig und numerische Felder rechtsbündig übertragen.

Ist Faktor 2 kleiner als das Ergebnisfeld, dann wird von alpha nach numerisch rechtsbündig und von numerisch nach alpha linksbündig übertragen.

Ist Faktor 2 größer als das Ergebnisfeld,dann wird von alpha nach numerisch linksbündig und von numerisch nach alpha rechtsbündig übertragen.

Die Servicefunktionen haben folgende Bedeutung:

ARR für ARRAY. MOVEV arbeitet in diesem Fall wie MOVEA.
LEF für Left. MOVEV arbeitet in diesem Fall wie MOVEL.
RIG für Right. MOVEV arbeitet in diesem Fall wie MOVE.

MOVEV ist nicht in den Einträgen indizierbar, kann aber wie folgt indiziert verarbeitet werden:

Beispiel:
```
- MOVE  X  TO  A
- MOVEL 'FG    ' TO A
- MOVEL 'RESULT' TO B
- MOVEV  A TO B  RIGHT
```

Damit wird das X-te Element der Feldgruppe FG in das Feld RESULT rechtsbündig übertragen.

Damit diese indizierte Verarbeitungsform möglich ist, müssen die beiden Einträge immer 10 Stellen große Alphafelder sein. In den ersten sechs Stellen dieser Felder steht immer der Feldname. Bei indizierter Verarbeitung steht also in den Stellen 1 bis 6 der Name der Feldgruppe, in den Stellen 7 bis 10 aber der Wert des Indexfeldes.

Servicefunktion Numerisch

Dieser Service ist nicht unterstützt für Feldgruppen oder Feldgruppenelemente.

1. Alpha nach numerisch: Übertragung wie bei MOVEN.

Beispiel:	Alpha (15)	Numerisch (9,3)
Feldinhalt:	123,999999	000123999C
Feldinhalt:	123,999999-	000123999D
Feldinhalt:	1234567,999999	234567999C
Feldinhalt:	-1234567,999999	234567999D

2. Numerisch nach alpha.

   - Die Übertragung erfolgt rechtsbündig.
   - Das empfangende Alphafeld muss groß genug sein.
   - Der Wert wird mit dem Edit-Code J aufbereitet.
   - Das Alphafeld sollte mit Blanks initialisiert werden.

Beispiel:	Numerisch (9,3)	Alpha (15)
Feldinhalt:	000123999C	123,999
Feldinhalt:	000123999D	123,999-
Feldinhalt:	234567999C	123.456,999
Feldinhalt:	234567999D	123.456,999-

Wird bei der Operation MOVEV ein Fehler festgestellt, z. B. dass der Feldname fehlerhaft oder der Index ungültig ist, so findet keine Übertragung statt. Es wird stattdessen der Schalter EF gesetzt, um den Fehler anzuzeigen.

MOVE-ARRAY Feldgruppeninhalt übertragen 4480

identisch mit MOVEA (s.o.)

MOVE-LEFT Feldinhalt linksbündig übertragen 4485

identisch mit MOVEL (s.o.)

MOVE-REST Rest einer Division übertragen 4490

identisch mit MVR (s.u.)

MOVE-RIGHT Feldinhalt rechtsbündig übertragen 4495

identisch mit MOVE (s.o.)

MULT Multiplizieren (RPG - Schreibweise) 4500

Zweck:

Zwei Werte sollen multipliziert werden.

Achtung: Diese Operation kann einfacher in Form einer arithmetischen Formel (C = B * A) geschrieben werden. Vergleiche hierzu die Operation '=' am Ende dieses Abschnitts.

Die Operation MULT wird syntaktisch genauso wie ADD behandelt. Entnehmen Sie Grammatik und Beispiele bitte sinngemäß der Beschreibung der Operation ADD.

MVR Divisionsrest übertragen 4505

(ON) OP EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP MVR oder MOVE-REST muss eingetragen werden
EG Feldname des Ergebnisfeldes

Beispiele:

- MVR REST

Zweck:

Retten eines Divisionsrestes.

Beschreibung :

Mit der Operation MVR kann (nur) unmittelbar nach einer Division der Rest in das in EG eingetragene Feld uebertragen werden.

MVR kann nur verwendet werden, wenn in der zugehörigen Division nicht gerundet und in EG keine Feldgruppe und kein Feldgruppenelement eingetragen wurde.

An EG wird automatisch eine Dezimalstellenanpassung vorgenommen.

Beispiel:
- QUOTIENT = 50 / 3
- MOVE-REST REST

Nach dieser Befehlsfolge hat das Feld REST den Inhalt 2.

OPEN Datei eröffnen 4515

(ON) OP F2 (SV) (BD)

ON Bedingungsabfrage (bis zu drei Schalter)
OP OPEN muss eingetragen werden
F2 Name der zu eröffnenden Datei
SV 'V' für variablen Dateinamen
BD Fehlerschalter

Beispiele:

- OPEN CPGWRK.
- OPEN DATEI VARIABEL

Zweck:

Eine Datei soll eröffnet werden.

Beschreibung:

Die in F2 angegebene Datei wird eröffnet.

Das interne Feld CPGFRC enthält nach dem OPEN den 'File Return-Code': 'NF' für 'not found in FCT'
'NO' für 'not open'
'EF' bei leerer VSAM-Datei **(Nur OPEN mit Service INPUT)**

Die Service-Funktion 'VARiabel' ermöglicht es, statt des expliziten Dateinamens in F2 ein (8-stellig alphanumerisch definiertes) Feld anzugeben, welches den Dateinamen enthält.

Die Servicefunktionen 'A', 'I' und 'R' sind für spezielle Anforderungen in HL1-Datasets unterstützt.

Es kann eine Ergebnisbezugszahl angegeben werden, die dann auf 'ein' gesetzt wird , wenn die Operation OPEN nicht erfolgreich abgeschlossen werden kann. Bei leeren VSAM-Dateien wird der Schalter EF gesetzt. Weiterhin wird das Feld CPGFRC mit 'EF' gefüllt.

Die Operation OPEN wurde erweitert. Ein OPEN für Input, Output, Update und Reuse für VSAM-Dateien im Batch wurde unterstützt. Bei der Ausführung kann man nur einmal ein OPEN mit Service durchführen.

Beispiel: OPEN CPGWRK INPUT

PARAMETER Parameterübergabe beim CALL 4517

PARM Parameterübergabe beim CALL 4518

OP F2

OP PARM oder PARAMETER muss eingetragen werden
F2 Feldname des Parameters

Beispiele:

- CALL 'PROGRAMM'
- PARM DATEI
- PARM SATZ

Zweck:

Eine Parameterleiste für den CALL-Befehl soll aufgebaut werden. Die erstellte Leiste entspricht den Konventionen für CALL-Schnittstellen.

Der Befehl ist nur sinnvoll nach einem Befehl CALL. Es können beliebig viele PARM-Befehle codiert werden, für jeden Parameter einer.

Zwischen CALL und PARM darf kein anderes Statement codiert werden.

PERFORM Unterprogramm ausführen 4520

identisch mit EXSR (s.o.)

PRNT Assembler-Listausgabe 4523

OP F2

OP PRNT muss eingetragen werden
F2 Konstanten 'ON', 'OFF', 'GEN' oder 'NOGEN'

Beispiel :

- PRNT NOGEN

Zweck:

Die Assembler-Listausgabe soll in der Procedure Division gesteuert werden.

Beschreibung:

Es hat sich gezeigt, dass der Programmierer normalerweise mit der Anlistung der TWA auskommt. Sollte sich jedoch die Notwendigkeit ergeben, Programmausschnitte anzulisten, so kann man die Assemblerliste in der Procedure Division steuern.

Die vier möglichen Eintragungen bedeuten im einzelnen:

PRNT ON : Ab diesem Statement soll die Assemblerliste gedruckt werden.

PRNT OFF : Ab hier soll keine Assemblerliste mehr gedruckt werden

PRNT GEN : Ab diesem Statement soll die Assemblerliste mit Makroauflösung gedruckt werden.

PRNT NOGEN: Ab hier soll die Assemblerliste ohne Makroauflösung gedruckt werden.

PROGRAM QPG-Programm ausführen (siehe Handbuch QPG) 4525

(ON) OP F2 (EG)

ON Bedingungsabfrage (bis zu drei Schalter)
OP PROGRAM oder PROGRAMM muss eingetragen werden
F2 Name des QPG-Programms, das im QTF hinterlegt ist
EG QTF-Library, falls das Programm nicht in PROG liegt

Beispiel :

```
PROGRAM  TESTOOP
```

Zweck:

Ein QPG-Programm soll ausgeführt werden.

Beschreibung:

QPG und der Befehl PROGRAM sind im QPG-Handbuch beschrieben.

PROG-VAR QPG-Programm ausführen (siehe Handbuch QPG) 4526

(ON) OP EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP PROG-VAR muss eingetragen werden
F2 Name eines 32-stelligen Feldes mit den Parametern

Beispiel :

```
PROG-VAR  QPGPAR
```

Zweck:

Ein QPG-Programm soll ausgeführt werden. Der Name des QPG-Programms und alle anderen für den Befehl relevanten Parameter sind in einem 32-stelligen Feld (variabel) abgestellt.

Das Feld QPGPAR hat folgenden Aufbau:

```
DATA DIVISION
     QPGPAR      0 * 32.    * Variable PROGRAM-Operation
     PROGN            8.    * Variabler Programmname
     PRLIB            4.    * Variabler Libraryname
     PREST           20.    * Reserviert
```

PROT(ECTION) Schutzcode vergeben (siehe Handbuch CPG3) 4527

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP PROT oder PROTECTION muss eingetragen werden
F2 Schutzcode direkt hexadezimal oder Feldname
SV VAR, wenn der Schutzcode einem Feld entnommen wird

Beispiel :

- PROTECTION FFFFF60E01

Zweck:

Ein Programm soll in Verbindung mit CPG3..Sign On vor unberechtigtem Zugriff geschützt werden.

Beschreibung:

Wird PROT mit variablem Protection-Code ausgeführt, also nach neuer Sign On-Logik, so wird ein 12-stelliges Alphafeld benötigt, das wie folgt aufgebaut ist:

Stelle 1 - 8 : Symbolischer Name des Schutzcodes
Stelle 9 : Art der Fehlerbehandlung
' ' - durch CPG3-Serviceprogramme
'R' - eigene Programmierung nach Abfrage des Return-Codes
Stelle 10 : Return-Code
0 - Zugriff ist berechtigt
1 - Non Terminal-Task, kann so nicht geschützt werden
2 - Benutzer ist nicht angemeldet
4 - Programm nicht in der Protection-Tabelle
7 - Fehler im Bereich 'Mandanten'
8 - Fehler im Bereich 'Sachgebiete'
Stelle 11 - 12 : Reserve

PURGE Temporary Storage Queue löschen 4530

```
-----------------------------------------------------
(ON) OP F2
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   PURGE  muss eingetragen werden
F2   Name der zu löschenden Temporary Storage Queue
-----------------------------------------------------

Beispiel :

- PURGE STOR
-----------------------------------------------------

Zweck:

Eine Temporary Storage Queue soll gelöscht werden.

Beachte:  Schalter  EF  wird beim PURGE gelöscht.
```

PUTIN Schalter zur nächst höheren HL1-Stufe übertragen 4535

```
-----------------------------------------------------
(ON) OP BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   PUTIN  muss eingetragen werden
BD   bis zu drei Schalter
-----------------------------------------------------

Beispiel :

- PUTIN  01 02 03
-----------------------------------------------------

Zweck:

Übertragen von Schaltern ( nur für HL1-Anwender ).

Beschreibung:

Mit der Operation PUTIN können bis zu drei Bezugszahlen
vom laufenden Programm in die  gleichlautende(n) Bezugs-
zahl(en) der  höheren  HL1-Stufe übertragen werden. Es
wird der  Zustand 'ein' oder 'aus'  unabhängig vom tat-
sächlichen aktuellen Zustand auf der höheren Stufe
übertragen.
```

PUTMI Schalter zur höchsten HL1-Stufe übertragen 4540

(ON) OP BD

ON Bedingungsabfrage (bis zu drei Schalter)
OP PUTMI muss eingetragen werden
BD bis zu drei Schalter

Beispiel :

- PUTMI 03 13 23

Zweck:

Übertragen von Schaltern (nur für HL1-Anwender).

Beschreibung:

Mit der Operation PUTMI können bis zu drei Bezugszahlen vom laufenden Programm in die gleichlautende(n) Bezugszahl(en) der höchsten HL1-Stufe übertragen werden. Es wird der Zustand 'ein' oder 'aus' unabhängig vom tatsächlichen aktuellen Zustand auf der höheren Stufe übertragen.

QSSA Qualifiziertes SSA aufbauen (DL/I-Anwender) 4545

```
-------------------------------------------------------
(ON) (F1) OP F2 EG LG (SV) OK (SV2)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Segmentname 8-stellig in Hochkommata
OP   QSSA muss eingetragen werden
F2   Suchargument 8-stellig in Hochkommata
EG   Name des Schlüsselfeldes
LG   Anzahl der linksbündig verwendeten Stellen des Key
SV   And, Or  oder  Variabel
OK   Vergleichsoperator ( EQ, NE, GE, LE, GT, LT )
SV2  '$' und bis zu drei DL/1 Commands ( D,F,L,N,Q,U,V )
-------------------------------------------------------

Beispiele:

- 'ARTIROOT' QSSA 'ART001  ' ATKEY 7   GT $ F
-            QSSA 'ART005  ' ATBEZ 1 O EQ

-------------------------------------------------------

Zweck:

Aufbau von qualifizierten SSAs.

Beschreibung:

In F1 wird der Segmentname 8-stellig in Hochkommata ein-
getragen. Dieser Eintrag  entfällt, wenn zur Angabe ei-
ner logischen  Verknüpfung in  SV ein  'A' für logisch
Und oder ein 'O' für logisch Oder eingetragen ist.

Der Service V bewirkt, dass die Werte der einzelnen Ein-
träge dem Feld CPGDLV entnommen werden.

In F2 wird ebenfalls  8-stellig in Hochkommata der  Name
des  sensitiven  Elements, d.h. des in  DL/I definierten
Sucharguments eingetragen.

EG enthält den Namen des  programmabhängigen  Schlüssel-
feldes. Aus diesem Feld werden die linken Stellen als
Schlüssel übertragen.

In LG wird die Anzahl der linksbündig verwendeten Stel-
len des Schlüsselfeldes in EG angegeben.
```

In OK wird ein Vergleichsoperator eingetragen:

Operator	Bedeutung
EQ	Suchargument ist gleich Schlüsselfeld
NE	Suchargument ist ungleich Schlüsselfeld
GT	Suchargument größer als Schlüsselfeld
LT	Suchargument kleiner als Schlüsselfeld
GE	Suchargument größer oder gleich Keyfeld
LE	Suchargument kleiner oder gleich Keyfeld

Zusätzlich kann der Befehl um bis zu drei DL/1-Commands erweitert werden. Hierzu wird hinter dem Operator ein $ (Dollar) angegeben und -jeweils durch Blanks getrennt- bis zu drei der folgenden einstelligen DL/I - Command-codes : L, F, D, N, Q, U, V.

Diese Command Codes sind in der Broschüre SH-12-5411 DL/I Application Programming, CALL and RQDLI Interface im Abschnitt 4 genau beschrieben.

RANDOM Datei freigeben 4550

identisch mit RNDOM (s.u.)

READ Lesen 4555

```
-------------------------------------------------------
(ON) (F1) OP F2 (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   READ  muss eingetragen werden
F1   Feldname des Schlüssels
F2   Dateiname
SV   Service: CLEar, LOW, A, C, K, S, T
-------------------------------------------------------
```

Beispiele:

- READ BILD
- KDNR READ KUNDEN.

Zweck:

Ein Satz einer Datei soll sequentiell gelesen werden.

Beschreibung:

1. Bildschirmdatei.

Mit dieser Operation wird ein Satz der in F2 genannten Datei gelesen. F1 bleibt bei hierbei frei.

Ein 'CLEar' in SV bewirkt, dass nach der READ-Operation der Schalter CL abgefragt werden kann. Andernfalls bewirkt die CLEAR-Taste die Beendigung des Programms.

Ein 'LOW' in SV bewirkt, dass Kleinbuchstaben nicht in Großbuchstaben übersetzt werden.

Ein 'S' schließt 'CLEAR' und 'LOW' ein.

Für 'quasi-transaktionsorientiertes' Programmieren stehen vier weitere Services zur Verfügung:

```
T   für die transaktionsorientierte Programmierweise
A   impliziert  T  und  LOW
C   impliziert  T  und  CLEar
K   impliziert  T,  LOW  und  CLEar
```

Achtung: Groß-/Kleinschreibung ist beim transaktionsorientierten Programmieren nur möglich, wenn in der TCT der Parameter UCTRAN nicht definiert wurde. Außerdem muss in der OPTIONS-Karte 'LOW' eingetragen sein.

Hinweis: Diese 'quasi-transaktionsorientierte' Programmierweise unterliegt einigen Einschränkungen. Sie darf nicht eingesetzt werden

- in Programmen, die mit EXPR oder EXITP mit Programmnamen aufgerufen werden
- in HL1 - Bausteinen
- wenn ein Dataset im Programm definiert ist

- wenn ein CHAIN für Update noch aktiv ist
  (CHAIN für Update wird dann ignoriert).

Sie sollte nur nach ausreichender Schulung angewandt werden.

2. Platten-Datei.

Plattensätze einer indexsequentiell organisierten Datei werden sequentiell gelesen. In F1 kann man den Feldnamen des Schlüssels angeben. Das Schlüsselfeld muss bei der erstmaligen Ausführung der Instruktion den Schlüssel des Satzes enthalten, mit dem die sequentielle Verarbeitung beginnen soll. F2 enthält den Namen der Datei.

Der Schlüssel kann generisch angegeben werden. Falls der eingetragene Schlüssel nicht in der Datei vorhanden ist, wird der Satz mit dem nächst höheren Schlüssel gelesen.

Bei Dateiende wird der Schalter 'EF' gesetzt und das interne Feld CPGFRC mit 'EF' gefüllt.

Bei VSAM-Dateien muss 'EF' nach der READ-Operation abgefragt werden, da bei einem weiteren READ nach EF das Programm mit einer Systemfehlermeldung abbricht.

3. Storage

F1 kann frei bleiben oder einen Schlüssel erhalten.

Im Normalfall wird der Bereich nach dem Lesen freigegeben. Ein 'S' oder 'SAVe' in SV bewirkt, dass der Bereich nach dem Lesen erhalten bleibt.

READB Datei rückwärts lesen 4565

```
-------------------------------------------------------
(ON) (F1) OP F2
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Feldname des Schlüssels kann angegeben werden
OP   READB  oder READ-BACK muss eingetragen werden
F2   Dateiname
-------------------------------------------------------
```

Beispiele:

```
- KEY READB DATEI.
- READ-BACK  KUNDEN
-------------------------------------------------------
```

Zweck:

Ein Satz oder mehrere Sätze einer VSAM-Datei sollen gelesen werden. Die Datei wird sequentiell rückwärts verarbeitet.

Beschreibung:

Die Operation READB für VSAM-Dateien arbeitet wie READ, jedoch werden die Sätze rückwärts gelesen. Das bedeutet, das der logisch nächste Satz der mit dem nächst kleineren Schlüssel ist. Bei Datei-Anfang wird der Schalter EF gesetzt und das interne Feld CPGFRC mit 'EF' gefüllt.

Unterschiedlich zum READ muss der erste mit READB gelesene Satz in der Datei vorhanden sein. Wenn der zu lesende Satz nicht vorhanden ist, so wird 'EF' gesetzt und keine Eingabe durchgeführt.

READB-PAGE Datei rückwärts in eine Seite einlesen 4570

```
-------------------------------------------------------
(ON) (F1) OP F2 EG
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Feldname des Schlüssels kann angegeben werden
OP   READB-PAGE  muss eingetragen werden
F2   Dateiname
EG   Name der Feldgruppe, in die eingelesen wird
-------------------------------------------------------
```

Beispiel:

```
- KEY READB-PAGE DATEI  FG
-------------------------------------------------------
```

Zweck:

Sätze einer VSAM-Datei sollen in eine Feldgruppe eingelesen werden. Die Datei wird sequentiell rückwärts verarbeitet.

Die Operation READB-PAGE ist eine Kombination der Operationen READ-BACK und READ-PAGE.

READI Segment einer Eingabedatei einlesen oder
Bildschirm transaktionsorientiert lesen 4575

```
-----------------------------------------------------------
(ON) OP F2 (DY) (EG)
oder
(ON) OP F2 (SV)
-----------------------------------------------------------
```

ON Bedingungsabfrage (bis zu drei Schalter)
OP READI muss eingetragen werden
F2 Name der Datei, aus der ein Segment gelesen wird
oder Name des Eingabebereichs in der Input-Division
DY Als Dummywort ist SEGMENT erlaubt.
EG Name des Segments, das im Input beschrieben ist
SV 'LOW' für Groß-/Kleinschreibung

```
-----------------------------------------------------------
```

Beispiel:

```
READI  CPGKDN  SEGMENT  STAMM
-----------------------------------------------------------
```

Zweck:

1. Aus einer bereits gelesenen Datei soll eine bestimmte Struktur nochmals ins Programm übertragen werden.
2. In einem transaktionsorientierten Programm ohne QSF-Maps soll eine bestimmte Eingabe angesteuert werden.

Beschreibung:

1. Übertragen von Segmenten

Insbesondere bei VSAM-Dateien mit verschiedenen Satzarten ist es interessant, zunächst einen Teil des Datensatzes zu lesen. Entsprechend dem Inhalt der gelesenen Daten entscheidet man dann, in welche Struktur der Eingabesatz übertragen wird.

Diese zusätzliche Übertragung bereits gelesener Eingabedaten erreicht man mit dem Befehl READI.

1.1. READI ohne Angabe eines Segments
Entsprechend den Eingabebestimmungen für die Datei werden die Eingabedaten nochmals übertragen.

1.2. READI mit Angabe eines Segments
Entsprechend den Eingabebestimmungen für ein Segment werden die Eingabedaten des zuletzt gelesenen Satzes nochmals übertragen. Segmente müssen unmittelbar hinter der Eingabedatei beschrieben sein, auf die sie sich beziehen.

Beachte:

READI ist immer nur nach einem READ-, READ-BACK-und CHAIN-U Befehl möglich. Wird diese Vorschrift nicht beachtet, so bricht das Programm bei der Ausführung ab.

READI ist jetzt auch für Temporary Storage unterstützt.

2. Taskorientierte Übertragung der Bildschirmeingabe

In transaktionsorientierten Programmen liest CPG automatisch bei Programmstart, also vor dem ersten Befehl der Procedure Division, vom Bildschirm die modifizierten Felder so ein, wie es in der Input Division für das zuletzt definierte Bild beschrieben ist.

Die Operation READI erlaubt es nun, gezielt eine ganz bestimmte Eingabe zu einem späteren Zeitpunkt anzusteuern.

Sollen beispielsweise aus programmiertechnischen Gründen Daten auf Temporary Storage zwischengespeichert werden, kann mit Hilfe von READI zuerst der temporäre Speicherbereich und danach der Bildschirminhalt eingelesen werden.

Die Servicefunktion 'LOW' verhindert, dass von CPG eine Übersetzung der Klein- in Großbuchstaben durchgeführt wird.

Achtung: Groß-/Kleinschreibung ist beim transaktionsorientierten Programmieren nur möglich, wenn in der TCT der Parameter UCTRAN nicht definiert wurde. Außerdem muss in der OPTIONS-Karte 'LOW' eingetragen sein.

READP Sätze einer Plattendatei in eine Seite einlesen 4580

```
-----------------------------------------------------
(ON) (F1) OP F2 EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   Name des Schlüsselfelds kann angegeben werden
OP   READP oder READ-PAGE muss eingetragen werden
F2   Name einer Platten-Datei
EG   Name einer Feldgruppe, in die eingelesen wird.
-----------------------------------------------------
```

Beispiel :

```
- KEY  READP  POSTEN  PAGE
- READ-PAGE  ARTSTA  S01
```

Zweck:

Sätze einer Datei sollen in eine Feldgruppe eingelesen werden.

Beschreibung:

Die Operation READP arbeitet wie READ. Das Schlüsselfeld in F1 gibt an, an welcher Stelle der Datei mit der Verarbeitung begonnen wird. Ist der Schlüssel nicht vorhanden, so wird beim nächst höheren aufgesetzt.

Es werden von der in F2 angegebenen Datei so viele Sätze gelesen, wie die in EG angegebene Feldgruppe Elemente hat.

Voraussetzung ist dabei, dass in der Output-Division beschrieben ist, wie die Feldgruppenelemente aufbereitet werden sollen.

Vor Beginn des Einlesens wird die Feldgruppe gelöscht.

READ-BACK Datei rückwärts lesen 4585

identisch mit READB (s.o.)

READ-PAGE Sätze einer Datei in eine Seite einlesen 4590

identisch mit READP (s.o.)

RECEIVE Transaktionsorientiertes Lesen einer QSF-Map 4595

identisch MAP (s.o.)

REPLACE Zeichen durch anderes Zeichen ersetzen 4600

REPLC Zeichen durch anderes Zeichen ersetzen 4601

```
-----------------------------------------------------
(ON) (F1) OP F2 EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
F1   zu ersetzendes Zeichen konstant oder variabel
OP   REPLC  oder REPLACE muss eingegeben werden
F2   einzusetzendes Zeichen konstant oder variabel
EG   Alphafeld / - feldgruppe / -feldguppenelement
-----------------------------------------------------

Beispiele:

-  REPLACE '0'  WERT.     * Blanks durch Nullen ersetzen
-  X'40' REPLC X'00' F10. * Blanks durch X'00' ersetzen
-  BYTE1 REPLC BYTE2 FG.
-----------------------------------------------------

Zweck:

Ein  beliebiges Zeichen in einem Feld soll durch ein an-
deres ersetzt werden. Im Defaultfall wird Blank ersetzt.

Beschreibung:

F1 enthält das Zeichen, das  ersetzt  werden  soll, als
alphanumerische oder hexadezimale Konstante oder als Va-
riable in einem einstelligen  Alphafeld.  Wird  F1 nicht
angegeben, so wird Blank ( X'40' ) ersetzt.

F2 enthält das Zeichen, das eingesetzt werden soll, als
alphanumerische oder hexadezimale Konstante oder als Va-
riable in einem einstelligen Alphafeld.

In EG steht der Name des zu verändernden Feldes.
```

RIGHT Alphafeld rechtsbündig verschieben, vorne Blanks 4605

identisch mit JRB (s.o.)

RIGHT-CHAR Alphafeld rechtsbündig verschieben, vorne Zeichen 4610

identisch mit JRC (s.o.)

RIGHT-ZERO Alphafeld rechtsbündig verschieben, vorne Nullen 4615

identisch mit JRZ (s.o.)

RNDOM Datei freigeben 4620

```
-----------------------------------------------------
(ON) OP F2
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   RNDOM  oder RANDOM  muss eingetragen werden
F2   Dateiname  oder  *ALL
-----------------------------------------------------

Beispiele:

-  RNDOM KUNDEN
-  RANDOM *ALL
-----------------------------------------------------
```

Zwecke:

- Eine Datei, die sequentiell verarbeitet wurde, soll auf Direktzugriff 'umgeschaltet' werden.

- Ein mit CHAIN UPDate gesperrter Satz soll wieder entriegelt werden, ohne dass ein Update erfolgt ist.

- Freigabe aller im Programm verwendeten Dateien (vgl. Kapitel 2300, Dateiverarbeitung)

- In einer Datenview soll auf den Anfang positioniert werden. Diese Positionierung ist z.B. erforderlich, wenn in einem Programm das Suchargument der Operation FIND wechselt und der interne Zeiger noch auf das zuletzt gefundene Tabellenelement zeigt.

ROLL Feldgruppe verschieben 4625

```
-----------------------------------------------------
(ON) OP EG
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   ROLL  muss eingetragen werden
EG   Name einer Feldgruppe
-----------------------------------------------------
```

Beispiel :

```
-  ROLL  PAGE
-----------------------------------------------------
```

Zweck :

Verschieben von Elementen einer Feldgruppe

Beschreibung:

Mit ROLL werden innerhalb einer Feldgruppe alle Elemente auf den nächst niedrigen Index verschoben. Das erste Element der Feldgruppe geht somit verloren; das letzte Element bleibt unverändert.

Beispiel : Inhalt der Feldgruppe FG

	vorher	nach ROLL FG
FG,1	(AAA)	(BBB)
FG,2	(BBB)	(CCC)
FG,3	(CCC)	(CCC)

Wird der Name der Feldgruppe mit einem Index versehen, so gibt dieser Index die Stelle innerhalb der Feldgruppe an, ab der gerollt werden soll.

Beispiel: ROLL-BACK FG(I) (Indexwert = 3)

Vorher:	Nachher:
AAAAAA	AAAAAA
BBBBBB	BBBBBB
CCCCCC	CCCCCC
DDDDDD	CCCCCC
EEEEEE	DDDDDD

ROLLB Feldgruppe rückwärts verschieben 4630

ROLL-BACK Feldgruppe rückwärts verschieben 4631

(ON) OP EG

ON Bedingungsabfrage (bis zu drei Schalter)
OP ROLLB oder ROLL-BACK muss eingetragen werden
EG Name einer Feldgruppe

Beispiel :

- ROLL-BACK FG

Zweck :

Verschieben von Elementen einer Feldgruppe

Beschreibung:

Mit ROLLB werden innerhalb einer Feldgruppe alle Elemente auf den nächst-höheren Index verschoben. Das letzte Element der Feldgruppe geht somit verloren; das erste Element bleibt unverändert.

Beispiel : Inhalt der Feldgruppe FG

	vorher	nach ROLLB FG
FG,1	(AAA)	(AAA)
FG,2	(BBB)	(AAA)
FG,3	(CCC)	(BBB)

Wird der Name der Feldgruppe mit einem Index versehen, so gibt dieser Index die Stelle innerhalb der Feldgruppe an, ab der gerollt werden soll.

Beispiel: Siehe Operation ROLL.

SAVET Bildschirminhalt retten 4640

```
-------------------------------------------------------
(ON) OP F2 (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SAVET  muss eingetragen werden
F2   4-stelliger Name einer Temporary Storage Queue
SV   VARiabel, falls der Name in einem Feld steht
-------------------------------------------------------

Beispiel :

- SAVET  TS03
-------------------------------------------------------

Zweck:

Das aktuell angezeigte Bild wird gerettet.

Beschreibung:

Die Operation SAVET rettet den  aktuellen  Bildschirmin-
halt in einen Temporary Storage Bereich.In F2 wird vier-
stellig der Name einer  TS-Queue angegeben,  auf die die
Bildschirmseite gerettet wird. ( Intern wird dieser Name
vorne um die vierstellige Terminal-Id erweitert. )

Der Name kann konstant vierstellig angegeben werden oder
in einem  Feld, das vierstellig alphanumerisch definiert
ist. Im zweiten Fall gibt man zusätzlich  VARiabel an.

Mit der Operation LOADT kann die gerettete Seite wieder
angezeigt werden.

Diese Operation kann benutzt werden, um aus einem Pro-
gramm heraus in ein anderes zu verzweigen und  hinterher
den alten Bildschirminhalt wiederherzustellen.

Beachte:

Das Zurückladen der Farben ist nur dann gewährleistet,
wenn sie per Farbattribut (B,G,P,R,T,W,Y) gesetzt werden
```

SAVET-VAR Bildschirminhalt retten 4642

wie SAVET und Servicefunktion VAR.

SCAN Alphafeld nach einer Zeichenfolge durchsuchen 4645

```
------------------------------------------------------
F1 OP F2 EG (BD) (SV)
------------------------------------------------------

F1   Suchargument (Zeichenfolge)
OP   SCAN  muss eingetragen werden
F2   Name des Feldes, das durchsucht werden soll
EG   Num. Feld für Startwert und gefundene Position
BD   Bezugszahl optional in der Form   # # BD
SV   Schlüsselwort V (Variabel)
------------------------------------------------------

Beispiele:

-  FELD  SCAN  FG(I) POS
-  FELD  SCAN  SATZ  POS V
------------------------------------------------------
```

Zweck:

Ein alphanumerisches Feld wird nach einer Zeichenfolge durchsucht.

Beschreibung:

Die Operation SCAN durchsucht das Feld in F2 nach dem Inhalt des Feldes in F1. Wird die Zeichenfolge gefunden, so wird im Ergebnisfeld die Position des ersten Zeichens von F1 im Feld F2 angegeben.

Als Ergebnisfeld kann ein numerisches Feld mit 0 Dezimalstellen angegeben werden. Ist dieses Feld vor dem Befehl größer 0, so wird erst bei der entsprechenden Position im zu durchsuchenden Feld mit der Suche begonnen. Nach der Operation enthält das Feld die Position, bei der die Zeichenfolge -d.h. das erste Zeichen der Folge- gefunden wurde. Wird die Zeichenfolge nicht gefunden, so wird das Ergebnisfeld wieder auf 0 gesetzt.

Falls mit einem solchen Indexfeld gearbeitet wird, kann auf Bezugszahlen verzichtet werden.

Die Operation SCAN ist in beiden Faktoren indizierbar.

Die Feldlänge von Faktor 2 muss größer als die Feldlänge von Faktor 1 sein. Bei Feldgruppen in Faktor 2 ist zu beachten, dass die Elementlänge größer sein muss als die Feldlänge in Faktor 1.

Als Schlüsselwort kann hinter dem Index noch ein V angegeben werden. Das zu vergleichende Feld wird dann nur mit dem im F2 angegebenen Zeichen und nicht mit der ganzen Feldlänge verglichen.

Eine vorgegebene Startposition wird berücksichtigt.

Mit dem Service 'V' kann das Suchen in variabler Länge ausgeführt werden.Die Länge ergibt sich durch die Anzahl

Stellen, die im Suchargument gefüllt sind. Das Ende des Arguments ist dabei Blank oder x'00'.Wenn nach einem Argument gesucht wird, das z.B. Leerstellen enthält, dann kann auch ein beliebiges Sonderzeichen benutzt werden,um das Suchargument einzuschließen (am Anfang und am Ende das gleiche Sonderzeichen). Das Sonderzeichen wird dabei nicht als Suchargument benutzt.

SCREENDUMP Testhilfe Special Terminaldump 4650

SDUMP Testhilfe Special Terminaldump 4651

(ON) OP (F2)

ON Bedingungsabfrage (bis zu drei Schalter)
OP SDUMP, TDUMP oder SCREENDUMP ist einzutragen
F2 4-stelliger Dump-Code zur Identifizierung

Beispiel :

- SCREENDUMP

Zweck:

Erzeugen eines Special Terminaldumps (vgl. Seite 2830) auf dem Bildschirm.

SELCT Feldauswahl 4655

SELECT Feldauswahl 4656

```
-------------------------------------------------------
(ON) OP EG (TYPE F2) (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SELCT oder SELECT muss eingetragen werden
EG   Name des Feldes, aus dem eingelesen wird
F2   Name der SELECT-Auswahl mit Schlüsselwort TYPe
SV   'INDex' , um die Indexprüfung zu unterdrücken
     oder 'VAR' für variablen SELECT-Namen
-------------------------------------------------------
```

Beispiele:

```
- SELCT CPGTCT
- SELECT FG(J) TYPE 01
-------------------------------------------------------
```

Zweck:

Einlesen von Feldern aus einem Feld.

Beschreibung:

Mit der Operation SELCT können über die Input-Division Teile alphanumerischer Felder oder Feldgruppenelemente in andere Felder übertragen werden.

Ist das Eingabefeld unterschiedlich strukturiert, so können verschiedene SELECT-Auswahlen spezifiziert werden. Die Eingabe kann dann mit dem Schlüsselwort TYPE in Verbindung mit einem frei wählbaren Namen direkt angesprochen werden. Selektiert wird dann nur, wenn die 'Typen' in Procedure und Input Division übereinstimmen.

Der SELECT-Typ kann variabel angegeben werden. Er wird dann einem Feld entnommen, das sechsstellig alphanumerisch definiert sein muss. An den Befehl wird dazu das Schlüsselwort VAR angehängt.

Beispiel :

```
- -I.   FIELD CPGCOM TYPE ALT
-        PAC  3  7 2  UMS
-       FIELD CPGCOM TYPE THAENS
-             3  8 0 KDNR
-            11 20   KURZN
:
- -C.   SELECT CPGCOM TYPE THAENS
```

In der Common Area (CPGCOM) stehen folgende Daten :

```
19002702 D LATTWEIN  DUEREN           **
```

Nach dem SELCT steht in KDNR der numerische Wert 2702
und in KURZN die Zeichenkette ' LATTWEIN '.

SEND Ausgabe einer QSF-Map auf den Bildschirm 4660

identisch mit MAPO

SETIN Schalter abhängig von einem Index setzen 4665

```
-----------------------------------------------------
(ON) OP EG (SV) BD
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SETIN oder SET-INDIC muss eingetragen werden
EG   Numerisches Indexfeld mit 0 Dezimalstellen
SV   'INDex',  um die Indexprüfung zu unterdrücken
BD   ein Schalter
-----------------------------------------------------
```

Beispiel :

```
- SETIN  INDEX  25
-----------------------------------------------------
```

Zweck:

Setzen eines Schalters in Abhängigkeit von einem Wert.

Beschreibung:

Der angegebene Schalter ist der erste in einer fortlaufenden Bezugszahlengruppe.

Nach Ausführung der Operation wird die Bezugszahl gesetzt, die sich aus der Addition der angegebenen Bezugszahl und dem Inhalt des Indexfeldes verringert um 1 ergibt. (Wäre also im obigen Beispiel der Wert von INDEX gleich 4, dann wäre nach dem SETIN der Schalter 28 gesetzt.)

Enthält EG keinen gültigen Index, so wird bei der Ausführung die Fehlermeldung 'INDEX FALSCH' ausgegeben.

Die Service-Funktion 'INDex' unterdrückt die Indexprüfung. Wenn das Indexfeld gleich Null ist, wird die Fehlermeldung 'INDEX FALSCH' nicht angezeigt und die Verarbeitung fortgesetzt.

SETIX Index abhängig von einem Schalter setzen 4670

```
-------------------------------------------------------
(ON) OP (F2) EG (SV) BD
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SETIX oder SET-INDEX  muss eingetragen werden
F2   Name einer Feldgruppe
EG   Numerisches Indexfeld mit 0 Dezimalstellen
SV   'INDex' zur Unterdrückung der Indexprüfung
BD   ein oder zwei  Schalter
-------------------------------------------------------
```

Beispiele:

```
- SETIX  INDEX  25
- SET-INDEX  FG  INTO  I
```

Zweck:

Wert eines Indexfeldes in Abhängigkeit von einer Bezugszahl setzen oder Ermitteln der Anzahl der Elemente einer Feldgruppe.

Beschreibung:

Der angegebene Schalter ist der erste in einer fortlaufenden Bezugszahlengruppe. Wird ein zweiter Schalter angegeben, so dient er der Begrenzung der Operation SETIX auf den Bereich von der ersten bis zu dieser Bezugszahl.

Nach Ausführung der Operation enthält das Indexfeld in EG den Wert, der sich aus der folgenden Rechnung ergibt: Ausgehend vom ersten Schalter der angegebenen Bezugszahlengruppe wird der nächste auf 'ein' gesetzte Schalter bestimmt. Von dessen Wert wird der Anfangswert der Gruppe subtrahiert und 1 aufaddiert.

Die Servicefunktion 'INDex' bewirkt, dass bei einem Indexfehler keine Fehlermeldung ausgegeben, das Indexfeld auf Null gelöscht und die Verarbeitung fortgesetzt wird.

Beispiele:

```
- SETIX  ZAHL  21
```

Ist die Bezugszahl 26 gesetzt und die Schalter 21 bis 25 sind aus, so enthält das Feld ZAHL nach der Operation den Wert 6.

```
- SETIX  FG  INDEX
```

Von der Feldgruppe FG wird die Anzahl der Elemente entsprechend der Data Division ermittelt und im numerischen Feld INDEX abgestellt.

SETLL Zeiger auf einen Satz einer Datei setzen 4675

(ON) F1 OP F2

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Feldname des Schlüssels, mit dem positioniert wird
OP SETLL oder SET-LIMIT muss eingetragen werden
F2 Dateiname

Beispiele:

- KEY SETLL CPGWRK

Zweck:

Auf einer Datei soll positioniert werden.

Beschreibung:

Mit dieser Operation kann bei sequentieller Verarbeitung die Reihenfolge unterbrochen und an beliebiger Stelle in der Datei wieder aufgesetzt werden.

Die SETLL-Operation liest keine Daten, sondern bestimmt durch den Inhalt des in F1 angegebenen Schlüsselfeldes lediglich den Satz, der mit der nächsten READ-Operation gelesen werden soll. F2 enthält den Namen der Datei.

Der Schlüssel in F1 muss auf der Datei nicht vorhanden sein. Es besteht also die Möglichkeit, mit einem Teilschlüssel zu positionieren. Es wird in einem solchen Fall beim nächst höheren Schlüssel aufgesetzt.

Ist bei der Ausführung einer SETLL-Operation das Ende einer Datei erreicht worden, so wird der Schalter EF gesetzt und das interne Feld CPGFRC mit 'EF' gefüllt.

Befindet sich die Datei nicht im sequentiellen Modus, so wird das interne Feld CPGFRC nach den Regeln des Befehls CHAIN gefüllt !

SETOF Schalter löschen 4680

(ON) OP BD

ON Bedingungsabfrage (bis zu drei Schalter)
OP SETOF muss eingetragen werden
BD bis zu drei Schalter können eingetragen werden

Beispiel :

- SETOF 01.

Zweck:

Bedingungs-Schalter sollen gelöscht werden.

Beschreibung:

Mit dieser Operation können bis zu drei Schalter von 01 bis 99 sowie T- und C-Schalter gelöscht werden.

SET(ON) Schalter setzen 4685

(ON) OP BD

ON Bedingungsabfrage (bis zu drei Schalter)
OP SET oder SETON muss eingetragen werden
BD bis zu drei Schalter können eingetragen werden

Beispiele:

- SET 10

Zweck:

Bedingungsschalter sollen gesetzt werden.

Beschreibung:

Mit dieser Operation können bis zu drei Schalter von 01 bis 99 sowie T- und C-Schalter gesetzt werden.

SET-INDEX Index abhängig von einem Schalter setzen 4690

identisch mit SETIX (s.o.)

SET-INDIC Schalter abhängig von einem Index setzen 4695

identisch mit SETIN (s.o.)

SET-LIMIT Zeiger auf einen Satz einer Datei setzen 4700

identisch mit SETLL (s.u.)

SORT(A) Feldgruppe sortieren 4705

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP SORTA oder SORT muss eingetragen werden
F2 Name einer Feldgruppe
SV 'Blanks', 'Descending'

Beispiele:

- SORT FG1.
- SORTA FG2 BLANKS-NACH-HINTEN

Zweck:

Eine Feldgruppe soll sortiert werden.

Beschreibung:

Die Operation SORTA sortiert die in F2 angegebene Feldgruppe nach Feldinhalt aufsteigend.

Die Servicefunktion 'Blanks' bewirkt, dass zusätzlich zur aufsteigenden Sortierung die Feldgruppenelemente, deren Inhalt blank (bzw. Null bei numerischen Feldgruppen) ist, nach hinten sortiert werden.

Die Servicefunktion 'D' bewirkt, dass die Feldgruppe in absteigender Reihenfolge sortiert wird.

Beispiel :

- SORTA FG1
- SORTA FG2 DESCENDING
- SORTA FG3 BLA

```
Feldinhalte  vorher  jeweils   G /T /1 /B /  /2 /L
Feldinhalt   nachher  FG1        /B /G /L /T /1 /2
Feldinhalt   nachher  FG2       2 /1 /T /L /G /B /
Feldinhalt   nachher  FG3       B /G /L /T /1 /2 /
```

SQARE-ROOT Quadratwurzel ziehen 4710

SQRT Quadratwurzel ziehen 4711

```
-----------------------------------------------------
(ON) OP F2 EG (SV)
-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SQRT oder SQARE-ROOT muss eingetragen werden
F2   Name eines numerischen Feldes
EG   Name eines numerischen Feldes ( für Wurzel )
SV   'H', 'RUNden', 'ROUnded'  zum  Runden
-----------------------------------------------------

Beispiele:

- SQRT ZAHL  WURZEL
- SQARE-ROOT F30    ROOT    RUNDEN
-----------------------------------------------------

Zweck:

Die Quadratwurzel einer Zahl soll gezogen werden.

Beachte :

Die Operation SQRT ist nicht indizierbar.

Mit den angegebenen Schlüsselworten in  SV ist es mög-
lich, das Ergebnis zu runden.
```

SUB Subtrahieren (RPG - Schreibweise) 4715

```
Zweck:

Zwei Werte sollen voneinander subtrahiert werden.


Achtung: Diese Operation kann  einfacher in  Form  einer
arithmetischen  Formel  (C = B - A)  geschrieben werden.
Vergleiche hierzu die Operation  '='  am Ende dieses Ab-
schnitts.


Die Operation SUB wird  syntaktisch  genauso wie ADD be-
handelt.  Entnehmen Sie  Grammatik und  Beispiele  bitte
sinngemäß der Beschreibung der Operation ADD.
```

SYNCP(OINT) Synchronization Point spezifizieren 4720

```
-------------------------------------------------------
(ON) OP (SV)
-------------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
OP   SYNCP oder SYNCPOINT muss eingetragen werden
SV   'ROL'  für Syncpoint Roll-Back
-------------------------------------------------------
```

Beispiel :

- SYNCPOINT

Zweck:

Syncronization Point spezifizieren.

Beschreibung :

Die Operation SYNCP gibt bei der System Recovery des TP-Steuerprogramms an, bis zu welchem Punkt die Verarbeitung abgeschlossen war. Alle nach einem SYNCP erfolgten Datei-Änderungen (Update, Hinzufügen, Löschen) werden nach einem anormalen Programm- oder Task-Ende über Dynamic Transaction Backout wieder rückgängig gemacht.

Voraussetzung für die Funktion der Operation SYNCP ist, dass für den TP-Monitor ein LOG-File angelegt wird und dass die FCT und die PCT entsprechende Eintragungen erhalten.

Für CICS lauten diese Eintragungen z.B. wie folgt :

FCT :	JID=System, LOG=YES	Für alle im Programm benutzten Update-Dateien
PCT :	DTB=YES	Für alle Programme, die SYNCP enthalten.

Mit der Servicefunktion ROLlback kann ein irrtümlich gegebener SYNCP-Befehl wieder aufgehoben werden.

Vor der Operation SYNCP müssen mit RNDOM *ALL alle im Programm verwendeten Dateien freigegeben werden.

TAG Merkmal setzen 4730

F1 (OP)

OP TAG kann eingetragen werden
F1 Merkmal (Anspringpunkt im Programm)

Beispiele:

- ANFANG TAG.
- ENDE-DER-RECHNUNG.

Zweck:

Eine Programmstelle soll einen Namen erhalten.

Beschreibung:

Diese Operation setzt ein Merkmal,zu dem mit Hilfe einer GOTO-Operation verzweigt werden kann. Der Name des Merkmals steht in F1.

Der Name des Merkmals darf nicht mit einem bereits verwendeten Feldnamen übereinstimmen.

Bei dieser Operation kann ausnahmsweise der Operationscode (TAG) entfallen. Ein einzelner Name in der Procedure-Division wird immer als TAG interpretiert.

TESTB Bit prüfen 4740

(ON) OP F2 EG BD

ON Bedingungsabfrage (bis zu drei Schalter)
OP TESTB oder TEST-BIT muss eingetragen werden
F2 zu prüfende Bits in Hochkommata
EG zu prüfendes, einstelliges alphanumerisches Feld
BD mindestens ein Schalter muss angegeben werden

Beispiele:

- TESTB '23' BYTE 10 11 12
- TEST-BIT '678' FELD # # 10

Zweck:

Einzelne Bits eines einstelligen, alphanumerischen Felds sollen abgefragt werden.

Beschreibung:

EG enthält den Namen des zu prüfenden Felds. F2 enthält in Hochkommata die zu prüfenden Bits in Form von Ziffern von 0 bis 7 in Zählrichtung von links nach rechts.

Die Ergebnisbezugszahlen werden nach Ausführung der Operation wie folgt gesetzt:

Wenn alle zu prüfenden Bits auf 'aus' gesetzt sind, wird die erste Bezugszahl gesetzt.

Sind die zu prüfenden Bits gemischt, d.h. zum Teil auf 'ein', zum anderen Teil auf 'aus' gesetzt, so wird die zweite Bezugszahl gesetzt. Die Anzahl der zu prüfenden Bits muss in diesem Fall größer als eins sein.

Sind alle zu prüfenden Bits auf 'ein' gesetzt, so wird die dritte Bezugszahl gesetzt.

Beispiel : - TESTB '23' BYTE 10 11 12

Inhalt des Feldes BYTE in Bit-/Hexschreibweise	gesetzter Schalter nach der Operation TESTB
1. (11110011) / (F3)	12
2. (11100011) / (E3)	11
3. (11000011) / (C3)	10

TESTF Feld auf numerische Zeichen prüfen 4743

```
-------------------------------------------------------
(ON) OP F2 (DY) EG (SV)
-------------------------------------------------------

OP   TESTF oder TEST-FIELD muss eingetragen werden
F2   zu prüfendes alphanumerisches Feld
EG   einstelliges Alphafeld für das Ergebnis von TESTF
SV   L , um auch das letzte Byte zu prüfen (Normalfall)
-------------------------------------------------------
```

Beispiele:

```
- TESTF FELD   STATUS  LAST
-------------------------------------------------------
```

Zweck:

Ein alphanumerisches Feld soll auf numerischen Inhalt untersucht werden.

Beschreibung:

Das alphanumerische Feld in F2 wird auf numerischen Inhalt überprüft. Das Ergebnis der Operation wird in EG abgestellt. Dabei bedeutet:

```
'B'  das Feld enthält nur Blanks (hexadezimal '40')
'M'  das Feld enthält nur Ziffern und führende Blanks
'N'  alle Zeichen sind Ziffern.
' '  wird in allen anderen Fällen gesetzt.
```

Im Normalfall wird die Servicefunktion L eingetragen, damit auch das letzte Byte auf Ziffer geprüft wird.

(In speziellen Fällen, in denen numerisch verarbeitete Felder ungepackt auf Dateien abgestellt wurden, arbeitet man ohne den Service L. In diesen Fällen kann die PACK-/UNPACK-Funktion des Assemblers dazu geführt haben, dass im letzten Byte des Feldes die Zone im ersten Halbbyte steht. Ist das eine Zone C oder D, dann stehen hier Buchstaben A bis I oder J bis R, die beim Packen wieder richtig in einen Wert umgesetzt würden. Auch solche Felder können mit TESTF getestet werden: Ohne Service L werden auch Buchstaben A bis R in der letzten Stelle wie die entsprechenden Ziffern erkannt und behandelt. Siehe Tabelle des EBCDIC-Codes.)

TESTN Feld auf numerische Zeichen prüfen 4745

```
-----------------------------------------------------
(ON) OP EG (SV) BD
-----------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   TESTN oder TEST-NUM muss eingetragen werden
EG   zu prüfendes alphanumerisches Feld
SV   'L', um auch das letzte Byte ( Zone F ) zu prüfen
BD   mindestens ein Schalter muss angegeben werden
-----------------------------------------------------
```

Beispiele:

```
- TESTN     FELD    10 11 12
- TEST-NUM  FELDA   #  10
```

Zweck:

Ein alphanumerisches Feld soll auf numerischen Inhalt untersucht werden.

Beschreibung:

EG enthält den Namen des zu prüfenden Felds.

Die Ergebnisbezugszahlen werden wie folgt gesetzt :

Die erste Bezugszahl wird gesetzt, wenn das Feld nur Ziffern enthält, die zweite , wenn das Feld außer den Ziffern auch führende Blanks enthält. Die letzte Bezugszahl wird gesetzt, wenn das Feld nur Blanks enthält

Die Servicefunktion 'L' bewirkt, dass auch das letzte Byte des Feldes auf Ziffer (Zone F, erstes Halbbyte) geprüft wird.

TESTT Bildschirmnamen prüfen 4750

```
-------------------------------------------------------
(ON) OP F2 BD
-------------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   TESTT oder TEST-TERM muss eingetragen werden
F2   Terminal-Name oder Namensteil in Hochkommata
BD   mindestens ein Schalter muss angegeben werden
-------------------------------------------------------

Beispiele:

- TESTT     'RZ'    10 11 12
- TEST-TERM 'SYS1'  #  #  10
-------------------------------------------------------

Zweck:

Ein  Terminal-Name oder -Namensteil  soll mit dem  Namen
des Terminals, an dem das Programm aufgerufen wurde,ver-
glichen werden.

Beschreibung:

Die Konstante in F2 wird mit der bei der Generierung der
TCT vergebenen  vierstelligen  Terminal-Identnummer ver-
glichen.

Ist die Konstante  kürzer als  vier Stellen, so wird in
dieser Länge linksbündig verglichen.

Die Bezugszahlen werden wie folgt gesetzt :

Ist der Name des aufrufenden Terminals  größer als der
Name in F2, wird der  erste Schalter gesetzt, der zweite
Schalter steht für den Vergleich auf kleiner, der drit-
für den Vergleich auf Gleichheit.

Beispiel:    - TESTT 'RZ'  10 20 30

Die gesamte Terminalgruppe, deren  Name mit  RZ beginnt,
würde den Vergleich erfüllen und Schalter 30 setzen.
```

TEST-BIT Bit prüfen 4760

identisch mit TESTB (s.o.)

TEST-FIELD Feld auf numerische Zeichen prüfen 4762

identisch mit TESTF (s.o.)

TEST-NUM Feld auf numerische Zeichen prüfen 4765

identisch mit TESTN (s.o.)

TEST-TERM Bildschirmnamen prüfen 4770

identisch mit TESTT (s.o.)

TIME Zeit setzen 4780

```
-----------------------------------------------------
(ON) OP (EG)
-----------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   TIME  muss eingetragen werden
EG   Name eines numerischen Feldes
-----------------------------------------------------

Beispiele:

- TIME
- TIME  MMSS.
-----------------------------------------------------

Zweck:

Die internen Felder UTIME und CPGTIM werden aktualisiert

Beschreibung:

In das in EG eingetragene Feld wird das 6-stellige nume-
rische Feld mit null Dezimalstellen CPGTIM rechtsbündig
übertragen.
```

TWALD TWA von Temporary Storage einlesen 4785

```
-------------------------------------------------------
(ON) OP F2 (SV)
-------------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   TWALD oder TWA-LOAD muss eingetragen werden
F2   Name eines Temporary-Storage-Bereichs
SV   VARiabel, falls der Name in einem Feld steht
-------------------------------------------------------
```

Beispiel:

- TWALD TS23

Zweck:

Eine mit TWA-SAVE auf Temporary Storage gerettete TWA wird wieder eingelesen.

Beschreibung:

In F2 wird ein Name für den TS-Bereich angegeben, entweder konstant vierstellig oder in einem Feld, das vierstellig alphanumerisch definiert ist. Im zweiten Fall muss zusätzlich das Schlüsselwort VAR codiert werden.

Für den TS-Bereich wird keine FILE-Bestimmung benötigt

Der Schalter EF wird gesetzt, wenn die TWA nicht gefunden wurde, außerdem wird das Feld CPGFRC mit 'EF' gefüllt.

TWALD-VAR TWA von Temporary Storage einlesen 4787

identisch mit TWALD und Servicefunktion VAR

TWASV TWA auf Temporary Storage retten 4790

(ON) OP F2 (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
OP TWASV oder TWA-SAVE muss eingetragen werden
F2 Name eines Temporary Storage Bereichs
SV VARiabel, falls der Name in einem Feld steht

Beispiel:

- TWA-SAVE TS23

Zweck:

Eine TWA wird auf Temporary Storage gerettet.

Beschreibung:

In F2 wird ein Name für den TS-Bereich angegeben, entweder konstant vierstellig oder in einem Feld, das vierstellig alphanumerisch definiert ist. Im zweiten Fall muss zusätzlich das Schlüsselwort VAR codiert werden.

Für den TS-Bereich wird keine FILE-Bestimmung benötigt

TWASV-VAR TWA auf Temporary Storage retten 4794

---

identisch mit TWASV und Servicefunktion VAR

TWA-LOAD TWA von Temporary Storage einlesen 4798

---

identisch mit TWALD (s.o.)

TWA-SAVE TWA auf Temporary Storage retten 4800

---

identisch mit TWASV (s.o.)

UCTRN Übersetzen in Großbuchstaben 4805

(ON) OP F2

ON Bedingungsabfrage (bis zu drei Schalter)
OP UCTRN muss eingetragen werden
F2 Eintragungen : ON oder OFF

Beispiel :

```
-  UCTRN OFF
```

Zweck:

Übersetzen der Eingabe in Großbuchstaben.

Beschreibung:

Der Eintrag ON in F2 setzt für den Bildschirm, an dem das Programm ausgeführt wird,das interne Feature UCTRAN auf ein. Damit übersetzt der TP-Monitor alle Eingaben von Kleinbuchstaben auf Großbuchstaben. Der Eintrag OFF setzt das Feature auf 'nicht übersetzen'.

Beispiel: UCTRN ON T bedeutet, dass bei Eingabe einer Transid eine Übersetzung erfolgt.

Somit ist es möglich, transaktionsorientiert auch Text-Eingaben in Groß-/Kleinschreibung zu erfassen. Die Operation COMRG bietet die Möglichkeit, den Status von UCTRAN abzufragen. Ist eine Anwendung beendet,sollte der Urzustand wiederhergestellt werden.

Auch CPG nimmt grundsätzlich eine Übersetzung vor. Über die OPTIONS-Bestimmung kann programmabhängig mit der Eintragung LOW das Einlesen von Text gesteuert werden; dies gilt aber nur für die interne Eingabeübertragung bei transaktionsorientierter Programmierung. Es kann für diesen Fall auch die Operation READI mit der Servicefunktion LOW verwendet werden.

Anwendungsbeispiel :

```
- -D.  TEST  32
- -I.  FILE  TEST  DS.  22  22  UCASE
- -C.  COMRG TEST.
-      IF  UCASE >< 'U'.  UCTRN  ON.  ENDIF
```

COMRG prüft, ob an diesem Bildschirm UCTRAN gesetzt ist; ist das nicht der Fall, wird das TCT-Feature mit der Operation UCTRN ON auf 'an' gesetzt.

UPDAT(E) Datensatz auf Datei/Storage-Queue zurückschreiben 4810

(ON) F1 OP F2 (EG)

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Name des Schlüsselfeldes
OP UPDAT oder UPDATE muss eingetragen werden
F2 Plattendatei/Storage-Queue, HL1-/QPG-Dataset
EG Name des Satzes (nur bei Plattendateien)

Beispiel :

- KEY UPDAT DATEI SATZ
- KEY UPDATE STOR
- UPDAT STOR

Zweck:

Ändern eines Satzes auf einer Plattendatei, einem HL1-Dataset oder einer Storage-Queue.

Beschreibung:

Es kann ein Satz auf einer Datei, einem HL1-/QPG-Dataset oder einem Storage verändert werden, ohne dass dabei die Ausgabe in der Output Division beschrieben werden muss.

Für Storage Queues und HL1-Datasets gilt:

In der Input Division dürfen sich nur alphanumerische und numerisch gepackte Felder befinden.

Für Plattendateien gilt:

Voraussetzung für die Durchführung der Operation ist, dass das in EG angegebene Feld über die Input Division unter der in F2 angegebenen Datei eingelesen wurde.

F1 enthält das Schlüsselfeld für den Dateizugriff, F2 den Namen der Datei, auf der der in EG angegebene Satz verändert wird.

Das Feld in EG muss alphanumerisch sein und darf maximal 256 Stellen lang sein. In der Regel muss es angegeben werden; bei der Verwendung von HL1-Datasets entfällt die Eintragung.

Für Dateien mit variabler Satzlänge ist die Operation UPDAT nicht unterstützt.

USSA Unqualifiziertes SSA aufbauen (DL/I-Anwender) 4815

(ON) F1 OP (SV)

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Segmentname 8-stellig in Hochkommata
OP USSA muss eingetragen werden
SV V für variable Einträge

Beispiel :

- 'ARTIROOT' USSA

Zweck:

Unqualifiziertes SSA aufbauen.

Beschreibung:

Zum Aufbau eines unqualifizierten SSA im DL/I muss nur der Segmentname 8-stellig in Hochkommata und der Operationscode USSA angegeben werden.

Nachdem alle SSAs gefüllt sind, kann der DL/I-Call erfolgen.

Der Service V bewirkt, dass die Werte der einzelnen Einträge dem Feld CPGDLV entnommen werden.

VBOMP VBOMP- oder EDN-Zugriff 4817

```
-----------------------------------------------------
(ON) (F1) OP F2 EG
-----------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
F1   Name des Schlüsselfeldes
OP   VBOMP muss eingetragen werden
F2   Name der Datei  oder  #
EG   VBOMP-/EDN-Operationscode
-----------------------------------------------------

Beispiel :

-  KEY  VBOMP  MASTER  MRAN
-----------------------------------------------------

Zweck:

VBOMP-Zugriff

Beschreibung:

Für VBOMP-/EDN-Verarbeitung stehen separate Merkblätter
zur Verfügung sowie die Literatur der Firma Wilken.
```

VSLCT VBOMP-/EDN-Daten in die TWA übertragen 4818

```
-----------------------------------------------------
(ON) OP F2
-----------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   VSLCT muss eingetragen werden
F2   Dateiname
-----------------------------------------------------

Beispiel :

-  VSLCT  MASTER
-----------------------------------------------------

Beschreibung:

Mit VSLCT werden die Daten aus der File Work Area in die
TWA-Felder übertragen.  Anders als bei anderen Dateizu-
griffen  bleibt diese  Area pro Datei  neben der TWA als
Datenbereich erhalten.  Der Anwender ist für die Über-
tragung der Daten in seine TWA selbst zuständig.

Für VBOMP-/EDN-Verarbeitung stehen separate Merkblätter
zur Verfügung sowie die Literatur der Firma Wilken.
```

WAIT Warten 4820

```
-----------------------------------------------------
(ON) OP (F2)
-----------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   WAIT  muss eingetragen werden
F2   Numerisches Feld für die Dauer des Wartens
-----------------------------------------------------
```

Beispiel :

- WAIT
- WAIT AMINIT

```
-----------------------------------------------------
```

Zweck:

Warten auf ein äußeres Ereignis.

Beschreibung:

Im ersten Beispiel wird 1 Sekunde gewartet.

Im zweiten Beispiel steht eine Uhrzeit im siebenstelligen numerischen Feld AMINIT wie folgt zur Verfügung: '0HHMMSS'.

WHEN Kennzeichnung einer Bedingung 4822

```
-----------------------------------------------------
OP F1 OK F2 (BO)
-----------------------------------------------------

OP   WHEN oder WHEN OTHER muss eingetragen werden
F1   Erstes Vergleichsfeld (indizierbar)
OK   Operator (siehe Tabelle der Operatoren)
F2   Zweites Vergleichsfeld (indizierbar)
BO   AND/OR zur logischen Verknüpfung
-----------------------------------------------------
```

Die WHEN-Anweisung kennzeichnet innerhalb einer EVALUATE-Operation eine alternative Bedingung.

Die WHEN-OTHER-Anweisung kennzeichnet innerhalb einer EVALUATE-Operation die Anweisung, welche durchgeführt wird, wenn alle vorhergehenden WHEN-Bedingungen nicht erfüllt waren.

(Siehe auch Kapitel strukturierte Programmierung, Operation EVALUATE und Beispiele 24/25)

Die Operation WHEN für die Alternativen im EVALUATE-Befehl kann analog zu IF und DO für Datumsvergleiche erweitert werden, also WHEN-DAT, WHEN-DATI, WHEN-DATK.

Beschreibung siehe IF-DAT, IF-DATI oder IF-DATK.

WRITE Datensatz auf Datei oder Storage neu anlegen 4825

(ON) F1 OP F2 (EG)

ON Bedingungsabfrage (bis zu drei Schalter)
F1 Name des Schlüsselfeldes
OP WRITE muss eingetragen werden
F2 Plattendatei/Storage-Queue/HL1-/QPG-Dataset
EG Name des Satzes (nur bei Plattendateien)

Beispiel :

- KEY WRITE DATEI SATZ
- KEY WRITE STOR
- WRITE STOR

Zweck:

Hinzufügen eines Satzes zu einer Plattendatei, einem HL1-/QPG-Dataset oder einer Storage-Queue.

Beschreibung:

Es kann ein Satz einer Datei, einem HL1-Dataset oder einer Storage Queue hinzugefügt werden, ohne dass dabei die Ausgabe in der Output Division beschrieben werden muss.

Für Storage-Queues und HL1-Datasets gilt:

In der Input Division dürfen sich nur alphanumerische und numerisch gepackte Felder befinden.

Für Plattendateien gilt:

F1 enthält das Schlüsselfeld für den Dateizugriff, F2 den Namen der Datei, auf der der in EG angegebene Satz verändert wird.

Das Feld in EG muss alphanumerisch sein und darf maximal 256 Stellen lang sein. Bei Plattendateien muss es angegeben werden.

Für Dateien mit variabler Satzlänge ist die Operation WRITE nicht unterstützt.

Voraussetzung für die Durchführung der Operation ist, dass das in EG angegebene Feld in der Input Division unter der in F2 angegebenen Datei beschreiben wurde.

XFOOT Summe einer Feldgruppe rechnen 4830

```
-------------------------------------------------------
(ON) OP F2 EG (SV) (BD)
-------------------------------------------------------

ON   Bedingungsabfrage  ( bis zu drei Schalter )
OP   XFOOT  muss eingetragen werden
F2   Name einer numerischen Feldgruppe
EG   Name des Summenfeldes
SV   'H', 'RUNden', 'ROUnded'  für Runden
BD   bis zu drei Schalter
-------------------------------------------------------
```

Beispiele:

```
-    XFOOT  FGN   SUMME
-    XFOOT  FGN   FELD   RUNDEN
-    XFOOT  MON   JAHR   #  01 01
```

Zweck:

Rechnen der Summe der Elemente einer Feldgruppe.

Beschreibung:

Nach Ausführung der Operation steht in EG die Summe der Elemente der in F2 angegebenen numerischen Feldgruppe.

Es besteht die Möglichkeit, Schalter zu setzen, die anzeigen, ob die Summe (in dieser Reihenfolge) größer, kleiner oder gleich null ist.

Mit der Service-Funktion kann bestimmt werden, ob das Ergebnis gerundet wird oder nicht.

Z-ADD Löschen und Addieren (RPG-Schreibweise) 4835

Zweck:

Einem numerischen Feld wird der Wert eines anderen numerischen Feldes zugewiesen.

Achtung: Diese Operation kann einfacher in Form einer arithmetischen Formel (A = B) geschrieben werden.

Die Operation Z-ADD wird syntaktisch analog zu ADD behandelt. Entnehmen Sie Grammatik und Beispiele bitte sinngemäß der Beschreibung der Operation ADD; die Eintragung in F1 entfällt.

Z-SUB Löschen und Subtrahieren (RPG - Schreibweise) 4840

Zweck:

Einem numerischen Feld wird der Wert eines anderen numerischen Feldes multipliziert mit -1 zugewiesen.

Die Operation Z-SUB wird syntaktisch analog zu ADD behandelt. Entnehmen Sie Grammatik und Beispiele bitte sinngemäß der Beschreibung der Operation ADD; die Eintragung in F1 entfällt.

Beispiel:

```
A=0-A
```

= Zuweisen von Werten und Rechenergebnissen 4850

```
-----------------------------------------------------
(ON) EG OP F1 (OK) (F2) (SV)
-----------------------------------------------------

-----------------------------------------------------

ON   Bedingungsabfrage (bis zu drei Schalter)
EG   Feldname des Ergebnisfeldes
OP   '=' oder ':='
F1   Feldname
OK   Operator: +, -, *, / für ADD SUB MULT DIV
F2   Faktor 2  (Name eines numerischen Feldes)
SV   Service: 'ROUNDed' oder 'RUNDen'  zum Runden
-----------------------------------------------------

Beispiele:
-    C = A + B.
-    PREIS = MENGE * FAKTOR.
-    X = K / 100.
-    CPGMCU = 'KDNR  '
-    FEHLER = 'Artikel nicht gefunden'
-    Z = 0.
-    GESAMTPREIS = GESAMTPREIS + WERT.
-    PI = 3,14.
-    X(I) = 0.
-    ON P1 X(I) = X(I) * 1,14 ROUNDED.
-    X := Y * FAKTOR.
-    FG(I)=-1*X.
-----------------------------------------------------
```

Zweck:

1. Eine einfache Formel nach den vier arithmetischen Grundrechenarten soll gerechnet werden.

2. Ein Wert soll zugewiesen werden. Die Operation ist sowohl für numerische als auch für alphanumerische und gemischte Operanden unterstützt.

Beschreibung zu 1.:

Die Formel darf nur in ihrer Grundform, bestehend aus Ergebnis und maximal zwei Faktoren verwendet werden. Schachtelformen sind nicht zulässig.

```
X = A + B
X = A - B
X = A * B
X = A / B
X = A
```

Für X = A + B gilt:
Nach Ausführung der Instruktion enthält das Ergebnisfeld (EG) die Summe von Faktor 1 (F1) und Faktor 2 (F2).

F1, F2 und EG können auch den Namen einer Feldgruppe enthalten. In diesem Fall werden alle Elemente dieser

Feldgruppe zu den entsprechenden Elementen der anderen Feldgruppe addiert.

Ist die Anzahl der Elemente unterschiedlich, so werden nur so viele Elemente addiert, wie für die kleinste Feldgruppe definiert wurden.

Wird der Name einer Feldgruppe eingetragen, so kann mit einem festen FG(3) FG(7) oder variablen FG(I) FG(KX) Index bestimmt werden,welche Einzelelemente der Feldgruppe zu addieren sind.

FG1(IN) = FG1(1) + FELDA.

Wobei FELDA der Name eines numerisch definierten Feldes, FG1 der Name einer numerisch definierten Feldgruppe, 1 der feste Index 1 und 'IN' der Name des numerisch definierten Indexfeldes ist, das den variablen Index enthält.

Für die anderen Grundrechenarten gilt sinngemäß das Gleiche.

Beschreibung zu 2.:

Beim Zuweisen von Werten steht links vom Gleichheitszeichen das Ergebnisfeld, rechts der zuzuweisende Wert als Variable oder Konstante.

Regeln:

- Sind beide Einträge numerisch, arbeitet die Operation wie Z-ADD.

- Sind beide Einträge alphanumerisch, wird wie beim MOVEL linksbündig geschoben. Konstante können bis zu 24 Stellen lang sein.

- Sind die beiden Einträge von unterschiedlichem Feldtyp, dann wird wie beim MOVE rechtsbündig geschoben.

H L 1 5000

HL1-Programmierung. 5010

HL1-Programme werden in der Programmiersprache CPG geschrieben. HL1 ist eine Erweiterung des CPG, die ab der Ausbaustufe CPG3 zur Verfügung steht.

Zur Generierung von HL1-Programmen ist der HL1-Compiler erforderlich, der mit EXEC HL1 aufgerufen wird. Dieser Compiler schließt CPG ein und kann über eine Eintragung in den OPTIONS auch 'normale' CPG-Programme generieren. Die erforderliche Eintragung ist MAIn oder ROOt für Hauptprogramm oder Root Phase. Diese beiden Einträge sind in ihrer Funktion identisch. (Also: HL1-Hauptprogramm = CPG-Programm)

HL1-Programmbausteine können aus jedem CPG-Programm und aus jedem HL1-Baustein mit der Operation 'EXECUTE HL1-Modul'

```
EXHM NAME
```

aufgerufen werden. Faktor 2 enthält dabei den Namen des HL1-Bausteins der verarbeitet werden soll.

HL1-Bausteine sind object-kompatibel, das heisst sie können ohne jede Änderung und ohne Neu-Umwandlung unter jedem TP-Monitor verarbeitet werden, für den ein CPG-Interface verfügbar ist.

Phasennamen für HL1-Programme sind bis zu 6 Stellen lang und frei wählbar.

Die Phasennamen für HL1-Bausteine werden vom Compiler links um die Konstante 'HM' erweitert, d.h. das Programm XZEIL1 wird unter dem Namen HMXZEIL1 katalogisiert. In der Core-Image-Library sind folglich alle HL1-Bausteine unter 'HM' zu finden, wodurch eine Unterscheidung von allen anderen Programmen gegeben ist.

HL1-Bausteine sind nicht selbständig ausführbar.

Der Aufruf erfolgt ausschließlich aus einem Hauptprogramm oder einem anderen HL1-Baustein mit Hilfe der Operation EXHM. Hauptprogramme werden ebenfalls durch HL1 erstellt (OPTIONS MAIn oder ROOt).

HL1-Programme sind reentrant geschrieben, das generierte Assembler-Quellenprogramm enthält nur reine Assembler-Instruktionen und ist frei von allen Ein- und Ausgabe-Makroinstruktionen und frei von Datenfeldern.

Alle Makro-Instruktionen werden in einem zentralen Interface zusammengefasst, das im Bedarfsfall eine schnelle Anpassungsmöglichkeit bei einem Wechsel der Basis-Software bietet und dadurch den Anwendungen einen hohen Grad von Flexibilität gibt.

Der Datenkanal 5020

Die Datenfelder eines CPG-Programms stehen in der Transaction Work Area (TWA), die bei der Online-Verarbeitung für jedes Terminal separat zur Verfügung steht.

Jeder HL1-Baustein hat zusätzlich eine eigene private Transaction Work Area (PWA - Private Work Area).

HL1-Bausteine sind unabhängig voneinander und können untereinander Daten austauschen.

Der Programmierer kann beim Aufruf eines HL1-Bausteins einen Datenkanal in die private TWA dieses Bausteins übertragen. Der Name dieses Datenkanals wird dazu hinter dem Modulnamen angegeben:

```
    EXHM MODUL KANAL1
```

Der Datenkanal wird in diesem Fall in den Eingabebestimmungen des rufenden Programms beschrieben:

```
-I.
    FILE KANAL1 HS DD.
       1   6 KDNR
       7  24 NAME
    P 25  26 0 NUMMER
```

Numerische Datenfelder sind im Kanal grundsätzlich gepackt !

Der Datenkanal muss im gerufenen Programm in den ersten Stellen der Data Division beschrieben sein. Es empfiehlt sich deshalb der Einsatz von Data Dictionary.

Die Felder KDNR, NAME und NUMMER werden dann beim Aufruf des Programms (EXHM) in die private TWA des Bausteins MODUL übertragen. Nach Ausführung des Bausteins MODUL werden die eventuell veränderten Feldinhalte wieder zurückübertragen.

Die Eintragung HS in der Input-Satzbesimmung bewirkt, dass für diese 'Eingabe'-Felder keine Optimierung durchgeführt wird.

Datenbeschreibung. 5030

Der Datenkanal wird im gerufenen Baustein am Anfang der Data Division beschrieben.

```
-D.
    DEFINE KANAL1
```

- Im Datenkanal sind numerische Felder immer gepackt !

- Der Datenkanal sollte keine Lücken enthalten, da im Kanal die Daten- anhand der Position, nicht anhand des Feldnamens übergeben werden !

Verarbeitung. 5050

HL1-Bausteine können von jedem CPG-Online-Programm und von jedem HL1-Batchprogramm aufgerufen werden.

HL1-Bausteine sind keine selbständig ausführbaren Programme. Sie sind in etwa vergleichbar mit Unterprogrammen, weil sie nur unter Steuerung eines übergeordneten Programms ablaufen können. Der Unterschied zu herkömmlichen Unterprogrammen besteht jedoch darin, dass sie nicht fest mit dem aufrufenden Programm verbunden sind, sondern je nach Bedarf erst zur Ausführungszeit zum Programm hinzu geladen werden.

HL1-Bausteine liegen daher nicht wie Unterprogramme als Bestandteile vieler Programme mehrfach in der Bibliothek für ausführbare Programme, sondern nur jeweils einmal.

HL1-Programme brauchen sehr viel weniger Platz in der Bibliothek als herkömmliche Programme.

Bei Online-Verarbeitung werden HL1-Bausteine wahlweise unter Steuerung des TP-Monitors oder in einen Programmpool der HL1-Methodenbank geladen. Der Programmpool wird selbsttätig initialisiert beim ersten Aufruf eines HL1-Bausteins. Alle weiteren Bausteine werden beim jeweils ersten Aufruf in die Methodenbank geladen.

Die Methodenbank ist eine zentrale Bibliothek für ausführbare Unterprogramme, auf die alle in der TP-Partition ablaufenden Programme zugreifen können.

HL1-Bausteine müssen daher nicht in die Programmtabellen des TP-Monitor-Programms eingetragen werden, jedoch ist eine Eintragung in einer HL1-Tabelle erforderlich. Bei dieser Eintragung handelt es sich jedoch nur um den Namen des Bausteins. Alle bei herkömmlichen Programmtabellen erforderlichen Parameter werden von HL1 selbst ermittelt.

VSE-Benutzer können ihre HL1-Bausteine von CICS verwalten lassen. Hieraus ergeben sich entscheidende Vorteile während der Testphase, aber auch wenn die Bausteine relativ groß sind. CICS versucht bei einem Newcopy diesen Platz wieder zu verwenden,falls das HL1-Modul nicht über eine 2K-Grenze verändert wurde. Der HL1-Aufruf und die Ausführung eines Bausteins werden hiervon nicht beeinflusst.

Besonderheiten bei der Datei-Verarbeitung mit HL1 5060

Grundsätzlich besteht bei der Dateiverarbeitung innerhalb des HL1-

Hauptprogramms und innerhalb von HL1-Bausteinen kein Unterschied zur Dateiverarbeitung im CPG.

Es gibt zwei Typen von Bausteinen: Standardbausteine und Dataset-Bausteine (OPTIONS DAT oder PWA).

Standardbausteine sollten nur dann eingesetzt werden, wenn sie ohne Dateizugriffe auskommen. Standardbausteine geben den benutzten Arbeitsbereich (PWA) nach jedem Aufruf wieder frei.

Dataset-Bausteine werden immer dann eingesetzt, wenn Dateien oder Datenbanken zu verarbeiten sind. Bei solchen Bausteinen bleibt die PWA (private Work Area) innerhalb der Task erhalten, d.h. die Inhalte aller Benutzerfelder stehen zur Verfügung (OPTIONS DATaset). Beim OPTIONS-Eintrag PWA werden die Benutzerfelder in der PWA initialisiert, nur die CPG-internen Felder bleiben erhalten.

Bei sequentieller Verarbeitung mit Dataset-Modulen verbessert sich die Performance sowohl online als auch im Batch. Es ist allerdings zu beachten, dass die PWA, die in diesen Modulen erhalten bleibt, im System Platz benötigt. Dies wird insbesondere bei Dialogprogrammierung bemerkbar sein. Außerdem belegen Dataset-Module bei READ oder CHAIN(U) jeweils eigene VSAM-Strings.

Private HL1-Bibliotheken. 5070

HL1-Bausteine werden nicht in die Programmtabellen der TP-Steuerprogramme eingetragen, weil sie von der HL1-Methodenbank gesteuert und verwaltet werden.

VSE-Benutzer können dennoch ihre HL1-Bausteine von CICS verwalten lassen. Hieraus ergeben sich Vorteile während der Testphase oder wenn es sich um besonders große Bausteine handelt. CICS versucht bei einem Newcopy, diesen Platz wieder zu verwenden, falls das HL1-Modul nicht über eine 2K-Grenze hinaus verändert wurde.

Jedoch ist eine Eintragung dieser Bausteine in einer HL1-Programm-Tabelle erforderlich. Die Eintragung erfordert je Baustein 4 Bytes und wird über den HL1-Tabellen-Generator (HL1HTG) generiert.

Die Größe der Programmtabelle ist auf 4K Bytes begrenzt. Eine HL1-Programmtabelle kann folglich nicht mehr als 1024 Eintragungen enthalten.

Die Reihenfolge der Eintragungen ist zwingend für die Verarbeitung. Eine spätere Änderung der Reihenfolge erfordert die Neu-Umwandlung aller Programme, die Bausteine aus dieser Tabelle aufrufen. Es ist daher sinnvoll, bereits vor der Installation eine Schätzung des späteren Bedarfs vorzunehmen. Werden insgesamt mehr als 1000 Bausteine für den Endausbau erwartet, so wird vorgeschlagen, bereits bei der Installation für die einzelnen Anwendungen private Bibliotheken anzulegen. Diese Bibliotheken bieten außerdem eine erhöhte Sicherheit bei Programmtests und bei der Programm-Wartung, da eventuelle Fehler auf die eigene Anwendung beschränkt bleiben.

Bei der Einrichtung privater Bibliotheken werden die Programm-Bausteine nach ihrer Verwendung sortiert. Sind die Bausteine eindeutig anwen-

dungsbezogen, z.B. Skonto rechnen, Artikeldaten lesen usw., so werden diese Bausteine in die private Bibliothek für diese Anwendung, z. B. Buchhaltung, Auftragsabwicklung usw., eingetragen. Sind die Bausteine anwendungsneutral, z.B. Programmbeschreibung, Bildschirmkopf usw., so werden diese Bausteine in die Tabelle für die allgemeine Bibliothek eingetragen, die zu jeder privaten Bibliothek hinzugefügt wird.

Private Bibliotheken werden durch einen einstelligen Suffix gekennzeichnet. Eine Muster-Tabelle für eine allgemeine Bibliothek und eine private Bibliothek mit dem Namen 'A' gehört zum Lieferumfang von HL1. Die beschriebenen HL1-Bibliotheken sind für eine allgemeine und maximal 23 private Bibliotheken ausgelegt.

Bei der Verwendung von Programmtabellen für private Bibliotheken muss bei einer HL1-Umwandlung die Spalte 22 der H-Karte unter Umständen eine entsprechende Eintragung enthalten. Siehe Abschnitt 9760.

Module, die zum Lieferumfang des HL1 gehören (z.B. die standardisierten Schnittstellen zu anderen Lattwein-Produkten) liegen in der privaten Library H.

HL1-Datasets 5080

HL1-Datasets sind eine Teilmenge der HL1-Bausteine. Grundsätzlich unterscheiden sie sich nicht von anderen HL1-Modulen.

HL1-Datasets eignen sich insbesondere zur dateiunabhängigen Programmierung, da sie mit gewohnten Dateioperationen ohne zusätzliche Programmierung zur Datei- oder Datenbankverarbeitung angezogen werden.

Bei HL1-Datasets sind folgende Punkte zu beachten:

- HL1-Datasets müssen nach der Dataset-Logik arbeiten, also in den OPTIONS mit den Parametern DAT oder PWA.

- Die Dateidefinition für ein HL1-Dataset hat als Einheit 'HL1' oder 'HL1DS' (und sollte im Data Dictionary hinterlegt sein).

- Die Datenübergabe vom rufenden zum gerufenen Modul und zurück kann über den HL1-Datenkanal sehr einfach abgewickelt werden. HL1-Datenkanäle enthalten nur alpanumerische und gepackte numerische Werte. Datenkanäle sollten keine Lücken enthalten und im Data Dictionary beschrieben werden. Sie werden im rufenden Programm im Input beschrieben, im gerufenen in den ersten Stellen der Data Division.

- Der Datenkanal muss das vierstellige Alphafeld CPGHIC (HL1 Interface Control) enthalten, das die Dataset-Verarbeitung steuert.

- Alle Dateioperationen sind für HL1-Datasets unterstützt.

- Der Programmierer, der mit einem HL1-Dataset arbeitet, benötigt lediglich Input- und Rechenbestimmungen. Dateien werden mit den Operationen UPDAT, WRITE und DELET geändert.

- Der Eintrag eines Schlüsselfelds bei der Dateioperation ist nicht erforderlich, da der Schlüssel im Datenkanal übergeben werden muss.

Programmierung mit HL1-Datasets 5090

Das folgende Beispiel zeigt ein HL1-Hauptprogramm mit den möglichen Dateizugriffen über ein HL1-Dataset.

```
    OPTIONS ROOT PHASE PROGRAMM.
    -F.
        FILE SAMPLE
    -D.
        FG 20 * 70
    -I.
        FILE SAMPLE HS DD.          * Data Dictionary generiert die
            1 14  KEY               * drei folgenden Statements:
           15 100 SATZ
          101 104 CPGHIC
    -C.
           CHAIN SAMPLE
           CHAIN SAMPLE CHECK.      * ohne Einlesen von Daten
           CHAIN SAMPLE UPDATE
           CHAIN SAMPLE P.          * Check für Update
           CHECK SAMPLE
           CLOSE SAMPLE
           DELETE SAMPLE
           OPEN SAMPLE
           RANDOM SAMPLE
           READ SAMPLE
           READ-PAGE SAMPLE FG.     * Seitenweise in eine Feldgruppe
           READ-BACK SAMPLE
           READB-BACK SAMPLE FG.    * Seitenweise rückwärts in eine FG
           SETLL SAMPLE
           UPDAT SAMPLE
           WRITE SAMPLE
```

Die Dateibeschreibung hat als Einheit HL1.

Für die Übertragung der Felder ist ein Datenkanal erforderlich. Dieser entspricht den Input-Statements bei Datei-Verarbeitung. Sein Name muss gleich dem Phasennamen des HL1-Dataset-Moduls sein. In der Input-Satzbestimmung wird das Schlüsselwort HS (für HL1-Struktur) angegeben.

Der Datenkanal beschreibt die Felder des Datasets. Zusätzlich muss er das 4-stellige HL1-Interface Controlfeld CPGHIC enthalten.

Datenkanäle sollten im Data Dictionary beschrieben werden !

Der File Return-Code wird in dem internen Feld CPGFRC zurückgegeben.

Das folgende Beispiel zeigt das HL1-Dataset-Modul für einen 1:1 Dateizugriff:

```
- OPTIONS DATASET PHASE SAMPLE.
-  FILE CPGWRK.
- -D.
-     DEFINE CPGWRK TYPE DS.      * Datenkanal wird wie folgt ge-
-                                 * neriert:
         KEY   14
         SATZ  86
         CPGHIC 4
-     ORG CPGHIC.                 * Redefinition von CPGHIC
-        OC  2.                   * HL1 Interface Opcode
-        RC  2.                   * HL1 Interface Return-Code
-        ORG.                     * Ende der Redefinition
- -I.
-     FILE CPGWRK DD TYPE DS.     * Dictionary generiert daraus:
          1  14 KEY
         15 100 SATZ
- -C.
-     EVALUATE
-       WHEN OC = 'B '.           * READ-BACK _____________________*
-         KEY READ-BACK CPGWRK
-         IF CPGFRC = 'EF'.       * End of File
-            MOVE 'E' TO RC
-         ENDIF
-       WHEN OC = 'C '.           * CLOSE _________________________*
-         CLOSE CPGWRK
-         IF CPGFRC >< '  '.      * Not found oder not closed
-            MOVE 'F' TO RC
-         ENDIF
-       WHEN OC = 'D '.           * RANDOM ________________________*
-         RANDOM CPGWRK
-       WHEN OC = 'G'.            * CHAIN _________________________*
-         IF OC = 'G '.           * CHAIN
-           KEY CHAIN CPGWRK
-         ELSE
-           IF OC = 'GU'.         * CHAIN für Update
-             KEY CHAIN CPGWRK UPDATE
-           ELSE
-             IF OC = 'GC'.       * CHAIN Check (ohne Einlesen)
-               KEY CHAIN CPGWRK CHECK
-             ELSE.               * CHAIN Check für Update
-               KEY CHAIN CPGWRK P
-             END
-           END
-         END
-         IF CPGFRC = 'NF'.       * Not found
-           MOVE 'G' TO RC
-         END
-       WHEN OC = 'L '.           * DELETE (Löschen) ______________*
-         EXCPT LÖSCHEN
-       WHEN OC = 'N '.           * WRITE (New Record) ____________*
-         EXCPT NEU
-         IF CPGFRC >< '  '.      * Duplicate Record
-           MOVE 'D' TO RC
-         END
-       WHEN OC = 'O '.           * OPEN __________________________*
-         OPEN CPGWRK
-         IF CPGFRC >< '  '.      * Not found oder not open
-           MOVE 'F' TO RC
-         END
-       WHEN OC = 'R '.           * READ __________________________*
```

```
-         KEY READ CPGWRK
-         IF CPGFRC = 'EF'.       * End of File
-           MOVE 'E' TO RC
-         END
-       WHEN OC = 'S '.           * SET LOWER LIMIT _____________*
-         KEY SETLL CPGWRK
-       WHEN OC = 'U '.           * UPDATE ______________________*
-         EXCPT ÄNDERN
-       WHEN OC = 'Z '.           * CHECK _______________________*
-         CHECK CPGWRK
-         IF CPGFRC =  '  '.      * !!!  Ausnahme !!!
-           MOVE 'G' TO RC
-         END
-     END-EVALUATE.               *_____________________________*
- -O.
-     FILE CPGWRK DD TYPE DS      ÄNDERN
-     FILE CPGWRK            DEL  LÖSCHEN
-     FILE CPGWRK DD TYPE DS ADD  NEU
```

Hinweise zur Programmierung

. Das Beispiel zeigt ein 1:1-Dataset, das beliebig um Logik erweitert werden kann.

  Hier ist natürlich der gesamte CPG-Funktionsumfang unterstützt, also auch der Zugriff auf eine beliebige andere Datenbasis. Der Programmierer des Anwendungsprogramms kann also auch bei anderer Datenbasis mit den gewohnte Lese- und Schreibbefehlen arbeiten.

. Datenkanäle sollten im Data Dictionary beschrieben werden, weil sie bei der Programmierung zweimal benötigt werden: Im Input des rufenden Programms und in der Datendefinition des gerufenen Programms.

  Bei 1:1-Datasets ist es oft der Fall, dass die Dataset-Struktur mit der Dateistruktur identisch ist, dass sie aber zusätzlich das CPGHIC-Feld enthalten muss. In diesem Fall kann an allen drei Stellen (Dataset-Kanal-Input im rufenden Programm, Kanalbeschreibung im gerufenen Modul und Datei-Inputbestimmungen im gerufenen Programm) mit der gleichen Data-Dictionary-Struktur gearbeitet werden. CPG generiert das CPGHIC-Feld dabei nicht im Input für die VSAM-Datei.

. Bei einem logischen Dateizugriff über ein Dataset können ca. 4000 Bytes verarbeitet werden.

. Die ersten Statements der Data Division müssen mit dem Datenkanal des rufenden Programms übereinstimmen. Felder dürfen unterschiedlich benannt sein, entscheidend ist die Position im Datenkanal.

  Das Interface-Controlfeld CPGHIC muss im Datenkanal liegen und steuert die Dataset-Verarbeitung: Die Art des Dateizugriffs wird über die ersten beiden Stellen des CPGHIC-Feldes an das Dataset-Modul übermittelt. Die 3. Stelle wird von CPG intern benutzt.

  Die 4. Stelle wird für den Returncode benutzt, der im rufenden Programm (automatisch) in das Feld CPGFRC umgesetzt wird.

Die Benutzung des HL1 Interface Controlfeldes CPGHIC ist in der fol-

genden Tabelle beschrieben.

CPGHIC Byte 1: Operationscode:	'B' READB 'C' CLOSE 'D' RNDOM 'E' EXCPT 'G' CHAIN 'L' DELET 'N' WRITE 'O' OPEN 'R' READ 'S' SETLL 'U' UPDAT 'Z' CHECK
CPGHIC Byte 2: Opcode Erweiterung:	erster Buchstabe der Servicefunktion, z.B. 'C' bei CHAIN CHECK
CPGHIC Byte 3: intern benutzt	
CPGHIC Byte 4: Returncode:	ist im HL1 Dataset zu definieren, dabei gilt: 'D' Duplicate Record 'E' End of File 'F' File Error bei Open und Close 'G' Satz nicht gefunden bei CHAIN oder CHECK

HL1-Batch-Programme 6300

Mit HL1 können auch Batchprogramme erstellt werden.

Dazu braucht in den OPTIONS nur BATch eingetragen zu werden.

Standardmäßig sind Batchprogramme für eine Programmgröße von bis zu 20K und eine TCA Größe bis zu 8K vorgesehen. Durch die OPTIONS-Parameter BIG und 12K können jedoch erheblich größere Programme erstellt werden.

Die Batch-Programme arbeiten genauso wie die meisten CPG- und HL1-Onlineprogramme mit einer Methodenbank. Die Batch-Methodenbank ist nicht Bestandteil des Programms, sondern wird zur Ausführungszeit zum Programm hinzugeladen.

Ebenso werden die Datenbereiche CSA, TCA und PIW (Interface Bereiche) beim Programmstart angefordert. Hiermit werden die HL1-Batch-Programme erheblich kleiner als vergleichbare RPG- oder COBOL-Programme. Dadurch wird der Platzbedarf in der Core Image Library wesentlich reduziert.

Zur Zeit sind im Batch folgende Dateiarten unterstützt:

READER für die Karteneingabe von SYSIPT.

PRINTER für die Druckausgabe auf SYSLST. Im Interface sind für die Druckausgabe zwei unterschiedliche DTF's verfügbar:

a) Standardmäßig DTFDI (Device independant), welches jedoch nur die Ausgabe bis zu 120 Stellen zulässt. Die Vorschubsteuerung erfolgt in diesem Fall über die L-Karte.
b) Außerdem DTFPR, welches eine Ausgabe bis zu 132 Stellen zulässt. Dieses DTF wird dann benutzt, wenn der Dateiname mit 'PRIN' beginnt.

Es können beliebig viele Drucker angegeben werden, die über die SYS-Nummer unterschieden werden.

PUNCHER für die Stanzausgabe auf SYSPCH.

Ein 'PUNCHER' wird jetzt erst bei der ersten Ausgabe eröffnet. Damit ist das Problem der 'leeren' Punch-Queue Einträge bei einigen VSE-Systemen mit Query Batch gelöst.

KSDS für VSAM-Indexdateien mit fester oder variabler Satzlänge.

Diese Angabe kann auch für VSAM-PATH-Dateien benutzt werden, wenn der Zugriff über einen Alternativindex erfolgt. Hierbei ist in der F-Karte der Pfadname einzutragen und die Keylänge vom Alternativindex anzugeben.

ESDS für sequentielle VSAM-Dateien mit fester oder variabler Satzlänge.

RRDS für VSAM-Satzadressdateien (haben immer feste Satzlänge)

DISK für sequentielle Plattendateien. Die Schlüssellänge in der

F-Karte muss leer bleiben.

TAPE für Banddateien.

DL1 für Datenbankzugriffe zu DL1.

HL1 für die Verarbeitung von HL1-Datasets.

STORAGE für die Verarbeitung von Temporary Storage. Es sind sowohl Einzelsätze als auch Queues unterstützt.

TABLE für die Verarbeitung von Tabellen.

Außerdem kann CPGTCT aus Kompatibilitätsgründen zu CICS-Programmen und -Modulen als Zwischenspeicherbereich verwendet werden.

Die Dateien werden bei Programmstart automatisch eröffnet und bei Programmende geschlossen, sofern nicht in der Files Division 'NO OPEN' angegeben ist. In diesem Fall wird die Datei erst per Programm mit den Befehlen 'OPEN' und 'CLOSE' eröffnet und geschlossen.

Bei dem Leser ist zu beachten, dass durch Abfrage von CPGFRC der Zustand 'End of File' geprüft werden muss. Lesen hinter End of File würde '/* ' oder '/& ' lesen und zu einem Systemfehler führen.

Ist ein Drucker angegeben, so kann durch den Druckernamen das benutzte DTF ausgewählt werden. Kanalvorschübe werden bei DTFDI über FORMS, bei DTFPR über FCB gesteuert. Ist ein Kanal nicht definiert, dann bricht das Programm mit einem I/O-Error ab. Zur Abfrage des Seitenüberlaufs kann der Overflow-Schalter 'OF' benutzt werden.

Der Schalter 'OF' wird gesetzt, wenn die Überlaufzeile überschritten wurde. Der Überlauf wird definiert durch die Größe des LIST-Dokuments (Anzahl Zeilen pro Seite) oder durch den Kanal 12 in der FORMS Division. Ist kein Kanal 12 angegeben, so wird in diesem Fall OF gesetzt, wenn die Formularlänge überschritten wurde.

Es wird empfohlen, dass die erste Druckausgabe einen Vorschub nach Kanal 01 enthält, da sonst die Steuerung mit dem 'OF' Schalter fehlerhaft sein kann.

Bei VSAM-Dateien wird im Data Dictionary oder im Programm in der FILES Division für die Ein-/Ausgabeart 'I', 'O' oder 'U' eingetragen werden. Bei 'I' wird die Datei nur für Eingabe eröffnet, bei 'O' und 'U' für Ausgabe. Das ist wichtig bei VSAM-Dateien mit Share Option 2. Dateien, die z.B. im CICS für Ausgabe eröffnet sind, können im Batch dann nur noch für Eingabe eröffnet werden. Ein Eröffnen für Ausgabe im Batch würde hierbei zu einem VSAM-Open-Fehler und damit zu einem Programmabbruch führen.

Für VSAM-Dateien ist die Option REUSE unterstützt. Damit können Arbeitsdateien erstellt werden, ohne dass hierzu immer ein DELETE/DEFINE erfolgen muss.

Die Dateiverarbeitungslogik ist im Batch die gleiche wie bei der Online-Verarbeitung.

Anders als bei Onlineverarbeitung können Sätze im Batch auch sequentiell upgedatet oder gelöscht werden.

Beim Hinzufügen von Sätzen ist folgendes zu beachten:

1. Sätze in KSDS-Dateien können direkt und sequentiell in Schlüsselfolge hinzugefügt werden. Bei sequentiellem Hinzufügen müssen die Sätze in aufsteigender Schlüsselfolge vorliegen. Sätze können zwischen bestehende Datensätze eingefügt werden. Existiert bereits ein Satz mit dem angegebenen Schlüssel, so wird 'Duplicate Record' im Feld CPGFRC und als Schalter 'DR' gemeldet.

2. Sätze in ESDS-Dateien werden sequentiell an das Ende der Datei angefügt. Es können keine Sätze zwischen bestehende Sätze eingefügt werden.

3. Sätze in RRDS-Dateien können sequentiell und direkt hinzugefügt werden. Bei sequentiellem Hinzufügen werden die Sätze an das Ende der Datei angefügt. Bei direktem Hinzufügen können Sätze nur eingefügt werden, wenn zu der angegebenen Satznummer kein Satz existiert, z. B. wenn der Satz vorher gelöscht wurde. Existiert bereits ein Satz mit der angegebenen Satznummer, so wird 'Duplicate Record' im Feld CPGFRC und als Schalter 'DR' gemeldet.

Beim Hinzufügen ist in der Ausgabe immer 'ADD' einzutragen. Dies gilt auch für das erstmalige Laden von Dateien.

Löschen von Sätzen ist nur in KSDS und RRDS Dateien möglich. Die Sätze können sowohl sequentiell als auch direkt gelöscht werden.

Eine Datei muss immer in dem Programm eröffnet werden, in dem sie verarbeitet wird, ansonsten wird bei der Ausführung ein (System-)Fehler gemeldet.

Einschränkungen und Hinweise zur Batch-Version 6310

Grundsätzlich sind im Batch alle CPG-Operationen unterstützt, bis auf folgende Ausnahmen:

1. Operationen, die speziell für Online-Verarbeitung konzipiert sind:

   COMRG DEBUG DEQ ENQ EREAD EXITD EXITI EXITT EXPR
   LOADT MAP MAPD MAPI MAPO MAPP PROT READI SAVET
   SDUMP SYNCP TESTT TWALD TWASV

   Ebenfalls ist die Operation RNDOM*ALL, die speziell für Online-Programmverbindungen konzipiert ist, nicht unterstützt.

2. Sonstige Operationen, die nicht unterstützt sind:

   CALLD CLEAR EXITP IFC

3. Die Operation CHAIN mit Service UPD oder 'P' sperrt den gelesenen Satz nicht für nachfolgende CHAIN-Operationen.

Folgende Schalter sind nicht unterstützt:

die Bildschirm-Schalter A1-A3 CL DE P1-PC Q1-QC SP und NI

Werden nicht erlaubte Operationen in einem Batchprogramm ausgeführt, so bricht das Programm mit einem formatierten Dump ab.

Module für Batch- und Online-Verarbeitung 6325

Ein Modul kann grundsätzlich sowohl online als auch im Batch eingesetzt werden. Sollte ein unterschiedlicher Ablauf zwischen Batch- und Onlineausführung erforderlich sein, so kann dies mit dem Feld CPGTID (Terminal Id) gesteuert werden:

```
- IF CPGTID = '    '.          * Blank im  Batch (X'40')
 :
- ELSE.                        * Online: Terminal-Id  oder
 :                             *         X'00' für Non-Terminal
- ENDIF.                       *               Tasks
```

Ausführen eines HL1-Batchprogramms 6330

Nach folgendem Muster kann der Batch-Job aufgebaut sein.

```
// JOB HL1BATCH
   ... hier ggfs. ASSGN, DLBL und EXTENT Karten einfügen
// EXEC BATCHPRG,SIZE=AUTO
   ... evtl. Datenkarten
/*
/&
```

Batchprogramme benötigen den GETVIS-Bereich der Partition. Daher ist in der EXEC-Anweisung immer der SIZE-Parameter anzugeben. Die Partition sollte ausreichend groß sein (z.B. 256K + evtl. benötigter Speicher bei Verwendung von Temporary Storage).

Sollten bei einer Programmausführung Fehler auftreten, so kann dies wie folgt verursacht sein:

1. Durch einen Fehler im Anwendungsprogramm, z. B. verursacht durch einen Datenfehler oder durch eine Operation die im Batch nicht unterstützt ist. Hierdurch bricht das Programm ab und es wird ein formatierter DUMP erzeugt. Die folgende Register- und Speicherbelegung hilft bei der Fehlersuche. In der Regel wird der Fehler im Datenbereich (Register 12) zu suchen sein.

2. Durch einen Fehler im Interface, z.B. einem VSAM-Fehler. Hierbei wird ebenfalls ein formatierter Dump gedruckt. Die Abbruchursache ist dabei am Anfang des Dumpbereichs im Klartext ausgedruckt. Gegebenenfalls sind VSAM-Error- und -Returncodes in der Broschüre 'VSAM Messages and Codes' nachzuschlagen. Die Angabe dieser Codes erfolgt dezimal. Nach einem VSAM-Fehler erfolgt ein Programmabbruch. Die folgenden Job-Steps werden dann nicht mehr ausgeführt.

UPSI-Schalter 6335

Bei VSE-Systemen können mit dem JCL-Statement // UPSI xxxxxxxx Schalter gesetzt werden. Diese Schalter können im Programm mit U1 bis U8 abgefragt werden.

UPSI-Schalter können mit SETON eingeschaltet und mit SETOF ausgeschaltet werden, dies gilt jedoch nur für die Dauer der Programmausführung.

Eventuell später laufende Jobsteps sind hiervon nicht betroffen, sie müssen gegebenenfalls durch weitere JCL-Statements gesetzt werden.

UPSI-Schalter können gesetzt werden um z.B. bestimmte Dateien von der Verarbeitung auszuwählen oder auszuschließen. In diesem Fall sind hierbei die Operationen OPEN und CLOSE zu verwenden.

Beispiel:

```
- IF CONDITION U1
-    OPEN DATEI1
- ENDIF
```

Abschnitt 6 Fehlermeldungen 6900

Fehlermeldungen während der Umwandlung 6900

Die Diagnostik arbeitet in 2 Schritten:

1. Fehlererkennung im Source-Code. Die Diagnostik wird u n t e r dem dem fehlerhaften Statements ausgewiesen.

2. Fehlererkennung im generierten Code. Die Diagnostik des CPG1-Compilers dokumentiert die Fehler ü b e r dem fehlerhaften Statement (über die OPTIONS wahlweise auch n e b e n dem Statement); wenn möglich, zeigt ein '$' die fehlerhafte Stelle an.

   Verschiedene Syntaxfehler können zu einer Fehlentscheidung des Compilers bei der Umwandlung führen, die eine weitere Diagnostik unsinnig macht. In diesen Fällen wird die Weiterverarbeitung abgebrochen und der Rest des Programms kommentarlos aufgelistet. Diese Beendung der Precompilierung beschränkt sich aber auf die OPTIONS sowie die Files-, Data- und Input-Division.

Fehlermeldungen in der Umwandlungsliste 6910

Alphafeld in Rechenoperation.

Bezugszahl nicht erlaubt.

Ursache: Es wurde eine Bezugszahl in der Output Division in der Satzbestimmung für eine Feldaufbereitung codiert.

Beispiel: - -O. FIELD SATZ ON 10

Aktion: EDIT-Aufbereitungen können nicht mit Schaltern verriegelt oder angesteuert werden. Schalter in den Feldbestimmungen sind erlaubt. Ansteuern einer bestimmten Aufbereitung ist möglich, wenn man in Procedure Division und Output Division mit dem Schlüsselwort TYPE arbeitet.

CPG-Fehler.

Alle CPG-Funktionen werden bei jedem Releasewechsel einzeln und in den in der Praxis häufig vorkommenden Kombinationen sorgfältig getestet. Bei aller Sorgfalt ist jedoch nicht auszuschließen, dass bei diesen Tests einmal ein Fehler nicht erfasst wird.

Aus diesem Grund führt der CPG eine Selbstdiagnostik durch, die dem Anwender die Fehlersuche ersparen soll, wenn ein solcher Ausnahmefall eintritt.

Die Meldung CPG FEHLER darf im Normalfall nie auftreten. Sollte dennoch bei einer Umwandlung diese Meldung erscheinen, so bitten wir, uns die Umwandlungsliste zur Prüfung zuzuschicken.

Datei doppelt definiert.

Die gleiche Datei wurde in den Files-Bestimmungen zweimal eingetragen. Der Dateiname und der Phasenname sind gleich.

Dateiname fehlt.

Die erste Ausgabebestimmung enthält keinen Dateinamen. Alle Folgekarten bis zur nächsten Satzbestimmung mit gültigem Dateinamen werden ignoriert.

Datei nicht definiert.

Es fehlt eine gültige Dateizuordnung.

Für die angegebene Datei fehlen die Input-Karten.

Wenn bei der Operation UPDAT und WRITE für die entsprechende Da- keine Eingabe (Input-Karten) definiert wurde.

Der Division-Indicator ist falsch.

Ursache: Es wurde ein syntaktisch falscher Indicator für den Beginn einer neuen Division angegeben.

Beispiele: Für die Output-Division wurde '- O.' statt '- -O.' codiert oder '- -OUTPUT' statt '- OUTPUT DIVISION'.

Aktion: Fehlerhaften Division Indicator verbessern

Die bedingenden Bezugszahlen sind unvollständig.

IF Condition wird mit anderen IF-Abfragen logisch verknüpft.

Aktion: Überprüfen der bedingenden Bezugszahlen und Korrektur. IF Condition kann zur Zeit nicht mit anderen IF-abfragen logisch verknüpft werden.

Die OPTIONS-Bestimmung ist nicht ordnungsgemäß abgeschlossen.

Ursache: Punkt, Semicolon oder das Schlüsselwort END fehlen als Abschluss der Options; Punkt ist angegeben, aber Semicolon wird erwartet.

Aktion: Richtige Satzende-Marke einsetzen.

Ein numerischer Wert wird erwartet.

Ursache: Es wurde ein alphanumerisches Wort an einer Stelle angegeben, an dem ein numerischer Wert erwartet wird.

Beispiel: In einer Feldbeschreibung der Input-Division wurde
- FELD 1 10 angegeben anstatt
- 1 10 FELD.

Aktion: Überprüfen und korrigieren des fehlerhaften Statements mit Hilfe der Syntaxregeln dieses Handbuch.

DO UNTIL / DO WHILE - FEHLER

Ursache: Bei einer logisch verknüpften DO-Schleife wurden DO-UNTIL und DO-WHILE gemischt verwendet.

Beispiel:
```
- DO WHILE ERRORCODE = '  ' AND
     UNTIL I > MAX.
```

Aktion: Entscheidung für DO-UNTIL oder DO-WHILE und Umformulierung der Schleifenbedingung, im Beispiel:
```
- DO WHILE ERRORCODE = '  ' AND
-    WHILE I <= MAX.
```

END Statement fehlt.

Nach einer DO- oder IF-Operation wurde kein gültiges END-Statement definiert.

Ergebnisfeld falsch.

Bei der CAB-Operation wurde kein Ergebnisfeld eingetragen.

EVALUATE kann nicht geschachtelt werden.

Ursache: Zwischen einem EVALUATE und dem zugehörigen END-EVALUATE darf kein weiteres EVALUATE codiert sein.

Falscher Operationscode in der Procedure Division

Ursache: Es wurde ein ungültiger Operationscode angegeben.

Aktion: Überprüfen der Opcode-Tabelle in diesem Handbuch und Operationscode verbessern.

Faktor 1 ist falsch

Beim CHAIN ist die Feldlänge von Faktor 1 ungleich der Keylänge aus der Files-Karte.

Beim READ eines Storage wurde Faktor 1 nicht mit 0 Dezimalstellen definiert.

Faktor 2 ist falsch.

Bei der LOKUP-Operation wurde ein fester Index eingetragen.

Faktor 2 keine Feldgruppe.

Der Feldname in Faktor 2 muss als Feldgruppe definiert sein.

Falsche Feldpositionen.

Bei der Eingabe-Feldbestimmung ist die Von- oder Bis-Position Null

Falsches oder fehlendes Schlüsselwort.

Ursache: Falsche Servicefunktion oder fehlendes 'INTO' bei CONVERT. In der Data Division Anzahl Dezimalstellen zu groß oder '*' bei der Feldgruppendefinition vergessen

Aktion: Korrigieren des fehlerhaften Statements.

Falsche Spaltenangabe in den Options.

Ursache: Mit dem Parameter COLumn in den Options wurde eine Spalte kleiner 7 oder größer 51 angesprochen.

Aktion: Überprüfen und korrigieren des Options-Eintrags mit Hilfe des Programmiererhandbuchs CPG1 oder HL1.

Feldaufbereitung fehlt.

Es fehlt in der Eingabe bzw. in der Ausgabe die Feldaufbereitung.

Feldgruppe zu groß.

Bei einer Diagrammausgabe wurde eine Feldgruppe angegeben, die insgesamt größer als 69 Stellen ist.

Feldname doppelt definiert.

Der gleiche Feldname wurde bereits für ein Feld mit anderen Felddaten (Feldlänge, Feldart, Dezimalstellen) vergeben.

Feldname fehlt.

Es wurde kein Feldname eingetragen.

Feld nicht definiert.

Ursache: Das Ergebnisfeld einer Zuweisung (mit =) ist im Programmcode bisher nicht definiert.

Aktion: Das Feld muss in die Data Division zur Definition aufgenommen werden.

Feld zu groß.

Ein alphanumerisches Feld ist größer als 256 Bytes, oder ein numerisches Feld wurde mit mehr als 15 Stellen generiert.

Logische IF-Verknüpfung unvollständig.

Ursache: Es wurde ein Gruppe von logisch verknüpften IF-Abfragen codiert, die unvollständig ist. Das heisst, dass einem AND oder OR kein IF mehr folgt.

Beispiel:
```
-  IF A = B  AND
-  IF B > C  AND.     * hier ist ein AND zuviel
-     VA = 'PRR '
-  ENDIF
```

Feld zu lang für binär.

Bei binärer Ausgabe darf das auszugebende numerische Feld maximal 9 Stellen groß sein. Bei größeren Feldern muss das Ergebnis vorher in ein 9-stelliges Feld zwischengespeichert werden.

Gruppenbegriff falsch definiert.

Es werden L-Schalter in den Calculations abgefragt, aber in den Input-Bestimmungen nicht definiert.

Hochkomma fehlt.

Eine Konstante wird nicht durch ein Hochkomma abgeschlossen.

HL1 Modul nicht gefunden.

Bei einer EXHM-Operation wurde ein Baustein eingetragen, der in der HL1-Tabelle nicht vorgefunden wird.

Index falsch.

Bei einer indizierten Operation wurde ein alphanumerisches Feld als Index eingetragen.

Keine Update Datei.

Eine Datei ist in den Files-Karten mit Input definiert, und in den Rechenbestimmungen wird versucht, die Operation UPDAT, DELET oder WRITE einzusetzen.

Label in Rechenoperation.

Es wurde ein Anspringpunkt anstelle eines gültigen Feldnamens eingetragen.

Nicht unterstützt.

Ursache: Zuweisung (mit =) einer alphanumerischen Konstanten zu einem numerischen Feld.

Zuweisung (mit =) einer alphanumerischen Konstanten, die länger als acht Stellen ist, zu einem Feldgruppenelement.

Halb-, Voll- oder Doppelwortgrenze für ein numerisches Feld.

Aktion: Überprüfen und korrigieren des fehlerhaften Statements mit Hilfe der Syntaxregeln dieses Handbuchs.

Numerische Eintragung zu groß.

Ursache: In der Data Division wurde ein Feld mit einer Länge größer tausend definiert.

Aktion: Korrigieren der Feldlänge, das Maximum ist 256.

P für Alphafeld.

Gepackte Ein- oder Ausgabe für ein Alphafeld ist nicht möglich.

Phasenname fehlt.

Es wurde kein Phasenname eingetragen.

Satzlänge fehlt.

In der Dateizuordnung ist eine Satzlängen-Eintragung erforderlich

Schablone für Alphafeld.

Ein Alphafeld kann nicht mit Schablone ausgegeben werden.

Schablone für Packfeld.

Ein numerisches Feld, welches gepackt ausgegeben wurde,kann nicht mit Schablone definiert werden.

Statement unvollständig.

Ursache: Ein Operationscode, z.B. LIST, erwartet weitere Einträge.

Aktion: Überprüfen und korrigieren des fehlerhaften Statements mit Hilfe der Syntaxregeln dieses Handbuchs.

Tabelle der erweiterten Namen ist voll.

Ursache: Der Precompiler kann aus erweiterten Feldnamen keine

CP-Namen mehr generieren.

Aktion: Vergrößern der Tabelle CPG*CETB durch den Systemprogrammierer.

Ungleiche Feldarten.

Bei der COMP-Operation wurden ungleiche Feldarten verwendet.

Ungültige Datei.

Es wurde kein gültiger Dateiname eingetragen.

Ungültige Dateiart.

Die Dateizuordnung enthält eine ungültige Ein-/Ausgabeeinheit.

Ungültige Einheit.

In der Dateizuordnung wurde eine ungültige Einheit angegeben.

Ungültige Länge

Ursache: Eine alphanumerische Konstante, die mehr als 26 Stellen lang ist, wurde versucht, mit der Operation = einem Alphafeld zuzuweisen.

Aktion: Fehlerhafte Konstante verkürzen oder statt der direkten Zuweisung eine Feldaufbereitung mit EDIT über die Output Division codieren.

Ungültige Programmfunktion

Gültige Programmfunktionen sind: P1 bis P9, PA, PB, PC, Q1 bis Q9, QA, QB, QC, A1 bis A3, DE, SP.

Ungültige Zeichen oder ungültige erweiterte Feldgruppenamen

Ursache:
- Bei Zuweisungen mit '=' ungültige Zeichen erkannt:
  - Zugewiesene numerische Konstante beginnt mit Dezimalkomma (bzw. Dezimalpunkt): Nicht unterstützt
  - Zugewiesene Alphakonstante beginnt nicht mit ' , sondern mit einem anderen Sonderzeichen, z.B. ".
  - Zuweisen einer hexadezimalen Konstante mit = ist nicht unterstützt.
- Bei einem erweiterten Feldgruppennamen wurde ein Name des Indexfeldes gewählt, der länger als eine Stelle ist.
- Zwischen Feldgruppe und Index steht ein Leerzeichen

Die vollständige Unterstützung erweiterter Feldgruppennamen ist nur dann gegeben, wenn die Options einen

Parameter LONg enthalten bzw. wenn die Spalte 100 der Standard-Header-Karte (CPGSTH) ein A enthält.

Ung. BREAK, CONTINUE, ELSE.

Bei der Operation ELSE wurde vorher keine gültige IF-Operation definiert. Bei der Operation BREAK oder CONTINUE wurde vorher keine gültige DO, DOU, DOW-Operation definiert.

Ungültiges END-Statement.

Es wurde vorher keine gültige DO bzw. IF-Operation definiert.

Ungültiges Farbattribut.

Es wurde kein gültiges Farbattribut eingetragen.

Ung. Feldlänge bei Code Y.

Der Aufbereitungsschlüssel 'Y' ist nur bei einer Feldlänge zwischen 3 und 6 mit null Dezimalstellen unterstützt.

Ung. Schablone/Konstante

Es wurde zu einem Schablonenkürzel auch noch eine Aufbereitungs-Schablone eingetragen.

Ung. Schablonenkürzel.

Es wurde kein gültiges Schablonenkürzel eingetragen,bzw. es wurde ein Schablonenkürzel bei einem gepackten Feld eingetragen. Bei Ausgabe eines gepackten Feldes ist die Ausgabe mit Schablone nicht zulässig.

Zu viele DO/IF Stufen.

Es sind maximal 40 DO/IF-Verschachtelungen möglich. Diese Anzahl wurde überschritten.

Zu viele Dateien.

Es wurden mehr als 100 Dateien in einem Programm definiert.

Zu viele Einträge im Statement.

Ursache: Ein Statement hat mehr Einträge, als für den benutzten Operationscode unterstützt sind.

Der Fehler tritt auch auf, wenn zwischen zwei Statements in einer Zeile kein Satzendezeichen steht. Der Fehler tritt oft vor Kommentarstatements auf.

Beachte: Als Eintrag wird möglicherweise auch eine Zeilennumerierung am rechten Rand erkannt, weil bis Stelle 79 gelesen wird. Abhilfe bei diesem Problem:

1. Jedes Statement mit einem Punkt beenden
2. Über die Standard-Header-Karte bestimmen, dass die Zeilennumerierung nicht als Programmcode berücksichtigt werden soll. In diesem Fall wird die Zeilennumerierung für die Zeit der Precompilierung mit Blanks überschrieben. Die Eintragungen in CPGSTH: 6 oder 8 für sechs- oder achtstellige Numerierung.

Aktion: Überprüfen und korrigieren des fehlerhaften Statements mit Hilfe der Syntaxregeln dieses Handbuch.

Zu viele Feldgruppen.

Es wurden mehr als 100 Feldgruppen in einem Programm definiert.

Zu viele Statements.

Ursache: Der Compiler findet ein Wort an einer Stelle, an der kein Statement mehr beginnen kann, z.B.
- bei der Codierung ab Stelle 8 steht ein Phasenname ab Stelle 75.
- bei der Codierung ab Stelle 1 darf die Codierzeile nur bis zur Spalte 72 genutzt werden.

Aktion: Überprüfen und korrigieren des fehlerhaften Statements oder Eintrag in den Standard-Header, um die letzten sechs oder acht Stellen der Zeile grundsätzlich nicht als Source-Code zu berücksichtigen.

Warnings 6920

Bei fehlerfreier CPG-Umwandlung ist es möglich, dass im Anschluss an die Umwandlung eine Warning ausgegeben wird. Es sollte dann anhand der unten aufgeführten Erklärungen des Warning-Codes überprüft werden, ob die Gefahr besteht, dass das Programm unerwünschte Ergebnisse liefert.

Warning: Ursache

CPG0010 - In einer Subroutine wurde ein (E)READ oder MAPx-Befehl mit einer Servicefunktion A, C, K oder T codiert.
- In einer Subroutine wurde ein EXITP oder EXPR-Befehl mit einer Servicefunktion IND oder SAVe codiert.

CPG0020 - Es wurden über 100 Listdokumente oder Datei-Einträge im Programm verwendet. Die Feldoptimierung sowie die Listreferenz kann nicht ordnungsgemäß ausgeführt werden.

CPG0030 - In einem Programm wurde in der Files Division ein Dataset definiert, das Programm arbeitet aber mit dem EXPR-Befehl und der Servicefunktion IND oder SAVe.

Wird im aufgerufenen Programm auch ein Dataset definiert, so kann die Warning ignoriert werden. Wird jedoch kein Dataset definiert, dann sind nach dem Rücksprung die Rettfelder für das Dataset überschrieben.

CPG0040 - Es wurde versucht, aus dem Data Dictionary eine Struktur anzuziehen,die im Data Dictionary File nicht angelegt ist.

Bei Warning CPG0040 wird der Umwandlungsjob mit CANCEL abgebrochen.

CPG0050 - Ein Feld wurde mehrfach definiert. Die Warning wird ausgegeben, wenn ein Dateiname als Feldname verwendet wird.

CPG0060 - In der Data Division ist eine Überlagerung nicht vollständig definiert.

CPG0061 - Diese Warning wird gesetzt, wenn ein Feld nicht definiert wurde.

CPG0070 - Das generierte Programm ist nicht ESA-Mode-fähig (siehe Kapitel 2970).

CPG0080 - Diese Warning wird gesetzt,wenn ein CP-Name aus einem Wort generiert wurde, das in den ersten fünf Stellen mit einem Opcode identisch ist, z. B. PURGETPTC.

CPG0090 - Diese Warning wird gesetzt,wenn ein OPTIONS-Parameter ohne Bedeutung gefunden wurde.

CPG0100 - Diese Warning wird ausgegeben, wenn bei EXCPT der Name in Faktor 2 mit ADD oder DEL beginnt.

Fehlermeldungen in der Assembly 6930

Werden bei der Umwandlung Assembler-Errors ausgewiesen, so handelt es sich in der Regel um Adressierungsfehler (Addressability Errors), die bei der CPG-Ausführung nicht festgestellt werden können. Das heisst, entweder ist das Programm oder die TWA zu groß.
Die Seite hinter der Programmierer-Check-Liste gibt Aufschluss über die fehlerhafte Programmroutine. (Siehe folgenden Listauszug).

```
                              EXTERNAL SYMBOL DICTIONARY        PAGE 1

SYMBOL    TYPE           ID  ADDR  LENGTH LD-ID

PROGN     SD (CSECT)    001 000000 005FFF


                              DUMMY SECTION DICTIONARY

SYMBOL     ID LENGTH

CPGTCADS  1FB 000FFF
```

Bei Standard-Programmen darf bei PROGN (Programmname) die Länge nicht größer X'5FFF' (24K) und CPGTCADS nicht größer X'FFF' (4K) sein.

Bei Programmen mit 8K TWA (OPTIONS ADD 1) darf die Programmgröße X'4FFF' (20K) nicht überschreiten und CPGTCADS nicht größer X'1FFF' (8K) sein.

BEI OPTIONS BIG oder 12K werden mehrere CSECTS generiert:

```
SYMBOL    TYPE           ID  ADDR  LENGTH LD-ID

PROGN     SD (CSECT)    001 000000 001FFF
BILD      SD (CSECT)    002 000000 002FFF
CPGF002   ER (EXTRN)    003
CPGF002   SD (CSECT)    004 000000 002FFF
SSSELCT   SD (CSECT)    005 000000 000FFF
BILD      ER (EXTRN)    006
SUBRN     SD (CSECT)    007 000000 002FFF
CPGAUSG   SD (CSECT)    008 000000 002FFF
PPEDIT    SD (CSECT)    009 000000 000FFF
```

Wurde BIG eingetragen, so darf bei PROGN die Länge nicht größer X'1FFF' (8K), bei Feldaufbereitung die Länge nicht größer X'FFF'(4K) und jede weitere CSECT (Subroutine) nicht größer als X'2FFF' (12K) sein.

Wurde 12K eingetragen, so darf die Programmlänge sowie Subroutine-Länge nicht größer X'1FFF' (8K) und CPGTCADS nicht größer X'2FFF' (12K) sein.

Fehlermeldungen bei der Ausführung 6950

Bei der Ausführung von CPG-Programmen kann am Bildschirm folgende Fehlermeldung auftreten:

```
**************************************************************************
SYSTEM-FEHLER                CPG 2.6 CLE        TT01  TST001    0002A0
**************************************************************************

DATEI CPGTST  NICHT ERÖFFNET

VERSTÄNDIGEN SIE BITTE IHR RECHENZENTRUM

**************************************************************************
PF1 ===> ABEND   +   PF3 ===> IGNORE   +   SONST ===> ENDE
**************************************************************************
```

Durch Betätigen der PF1 - Taste wird das Programm abgebrochen und ein CICS-Dump erzeugt. Durch Betätigen der PF3-Taste erfolgt ein Rücksprung ins Anwendungsprogramm.

Diese Auswahl kann bei der CPG-Installation eingeschränkt werden.

Beim Betätigen einer anderen Funktionstaste wird das Programm ohne Dump beendet. Über die Kundenkonfiguration kann gesteuert werden, ob jedesmal ein Dump und abnormales Ende des Programms gesetzt wird (wegen DTB). Der Dump-Code und die Fehlermeldung am Bildschirm geben Auskunft über die Fehlerursache. Folgende Fehlermeldungen können am Bildschirm auftreten.:

Datei XXXXX Bereich zu klein:

- Beim Hinzufügen von Datensätzen wurde kein freier Plattenspeicher mehr gefunden.
- Es kann keine Druckausgabe mehr durchgeführt werden, es ist kein freier Transient-Data-Bereich mehr verfügbar.

Datei XXXXX Eingabefehler:

- Das System kann auf die entsprechende Einheit nicht zugreifen. (Hardware Defekt).
- Bei einer Druckausgabe wurde versucht, einen Satz auszugeben, dessen Länge größer ist als die Satzlänge des VSAM-Clusters DFHNTRA.

Datei XXXXX nicht eröffnet:

Die Datei wurde für die Online-Verarbeitung nicht eröffnet. Ein CICS-Dump ist bei dieser Fehlermeldung nicht möglich.

Datei XXXXX nicht in der FCT:

Es wurde eine Ein- / Ausgabe mit einer Datei durchgeführt, die in der CICS-FCT nicht definiert ist.

Datei XXXXX nicht verfügbar:

- Es wurde ein Update versucht, ohne vorher ein CHAIN für diesen Satz durchgeführt zu haben. (Put no Get).

- Es wurde versucht, einen nicht existierenden Satz mit Update zu verändern.
- Bei CICS 1.6 kann diese Meldung auch erscheinen, wenn bei einem WRITE kein vorheriges CHAIN durchgeführt wurde.
- Die Dateikomponente zwischen Define Cluster und CICSFCT stimmt nicht überein.
- Der Service für diesen Zugriff ist im CICSFCP nicht generiert (Reg 9 = X'00') VSAM und ISAM im FCP nicht unterstützt.
- Der Service für diesen Zugriff ist in der CICSFCT nicht generiert. (Reg 9 ungleich X'00' VSAM Delete und Delete nicht als SERVREQ aufgeführt).
- Es wurde ein SYNCP gegeben, ohne vorher die Dateien mit RNDOM freizugeben.

Datei XXXXXXXX Programmfehler Error Code

Es soll eine Operationsfolge (Dateizugriff) durchgeführt werden, die der TP-Monitor untersagt. Folgende Operationen dürfen online nicht hintereinander für die gleiche Datei ausgeführt werden:

Error Code	Operation	Auszuführende Operation
SEQ FU	Nach einem READ	darf kein UPDATE durchg. werden
SEQ FU	" " READ	" " CHAIN UPD " "
SEQ FA	" " READ	" " ADD " "
SEQ FL	" " READ	" " DELET " "
SEQ EF	" " EF	" " READ " "
UPD FC	" " CHAIN UPD	" " CHAIN " "
UPD FR	" " CHAIN UPD	" " READ " "
UPD FB	" " CHAIN UPD	" " READB " "
UPD FU	" " CHAIN UPD	" " CHAIN UPD " "
UPD FA	" " CHAIN UPD	" " ADD " "

Sollten in Ihrem Haus diese Regeln nicht eingehalten worden sein, so kann für eine Übergangszeit diese Fehlermeldung unterdrückt werden.

In diesem Fall wird eine entsprechende Fehlermeldung auf der Systemkonsole ausgegeben und die Programme sollten schnellstens umgestellt werden.

Beispiel einer solchen Fehlermeldung auf der Konsole:

```
-------------------------------------------------------------
F2 002 *** FILE  CPGWRK  PROGRAM-ERROR     UPD FL
F2 002 *** TRANSID TT02  PROGRAM TST002   TERM RZR1    ****
-------------------------------------------------------------
```

Datei XXXXX VSAM Error Illogic:

Diese Fehlermeldung erscheint, wenn ein VSAM-Fehler auftritt, der keinem anderen Fehlercode entspricht.
Der VSAM-Error und -Return-Code wird am Bildschirm mit angezeigt oder kann im CICS Transaction Dump im Transaction Storage File (Länge = D0) auf Displacement 2C nachgesehen werden. (z.B. 0008006C = Returncode 8 und Errorcode 6c).
Für Alternativindex-Verarbeitung traten zuviele gleiche Keys auf, so dass die Satzlänge des VSAM-AIX zu klein war.
VSAM Error/Return-Codes können in der IBM Broschüre VSE/VSAM Messages und Codes (SC 24-5146) nachgesehen werden.

Division durch 0:

Bei der Operation DIV hatte der Divisor (Faktor 2) den Feldinhalt Null.

Divisions Fehler:

Bei der Operation DIV wurden die Regeln für die Feldgröße nicht eingehalten.

Entscheidungstabellen-Verarbeitung

Wenn immer nur die erste Aktion durchgeführt wird,ist evtl. eine Numerierung im CPG-Programm ab Spalte 73 vorgenommen worden.

ESA-Fehler

Aus einem ESA-Mode-fähigen Programm (siehe Kapitel 2970) wurde ein HL1-Modul aufgerufen, dass nicht mit einem ESA-fähigen CPG (Releasestände kleiner 1.6) umgewandelt wurde.

HL1 Fehler Datei: XXXXXXX HMModul

Beim EXHM wird festgestellt, dass ein Modul aufgerufen wird, das nicht nach der Dataset-Logik arbeitet und das Dateien enthält, die im übergeordneten Hauptprogramm nicht definiert sind.

Der Fehler wird behoben, indem man in den Options einen Parameter für Dataset-Verarbeitung einträgt (z.B. DATaset oder PWA).

HL1 Pool Fehler

Beim EXHM wird festgestellt, dass entweder der HL1-Pool voll ist oder dass ein HL1-Modul nicht ordnungsgemäß ausgeführt werden kann (z.B. weil eine private Library zur Ausführungszeit nicht gefunden wurde). Zur Fehlerbehebung siehe CPG-Handbuch, S. 9770

Index falsch:

Es wurde eine Operation mit variablem Index ausgeführt.
Der Inhalt des Indexfeldes ist entweder Null oder größer als die Anzahl der definierten Elemente der entsprechenden Feldgruppe, bzw. bei SETIX oder SETIN ist der Index zu groß.

READI Fehler

Ein READI wurde ausgeführt, ohne dass vorab ein READ oder READB ausgeführt wurde.

SQRT negativ:

Bei der Operation SQRT wurde ein negativer Feldinhalt vorgefunden.

TSQUE XXXXXXXX NNN TCA RCODE:

Bei der Ausgabe einer Temporary Storage Queue traten Fehler auf.
XXXXXXXX gibt den Namen des Storagebereichs an, auf den ausgegeben werden sollte.

NNN gibt den Temporary Storage Return-Code (dezimal) an.

(Nähere Einzelheiten siehe CICS-Broschüre DFHTS TYPE=PUTQ).

TWALENGTH: xxx/ yyy

Beim quasi-transaktionsorientierten Programmieren wurde festgestellt, dass die TWA-Länge yyy nicht der Länge der geretteten TWA xxx entspricht. Gesteuert über CPGURSI2, Stelle 7.

Der Dump-Code aus dem CICS-Dump gibt über die Fehlerart Auskunft. Folgende Codes können auftreten:

'CPGL' Das Command Level Interface ist nicht geladen.

'ECPG' Allgemeine CPG-Fehlermeldungen

'ECPS' Storage Control. Beim Initialisieren einer Task war zu wenig Speicher verfügbar.

'ECPT' Terminal Control. Bei einer Bildschirmein- oder -ausgabe war kein Speicher verfügbar.

'ECPW' Die Eintragung TWA-Size in der CICS-PCT wurde zu klein eingetragen.

Bei HL1 Modulen private TWA (PWA) durch Kanal überschritten.

'NDLI' DL/I-Anwender. In der OPTIONS DIVISION fehlt die Eintragung ADDressierung D.

'QSFI' Bei einer MAP-Operation wurde die angegebene Map im QSF nicht gefunden.

'QSFM' Fehler in der Directory oder Phase nicht gefunden.

'QSFS' Es liegt ein Size Error vor.

'QSFT' In einer Non-Terminal-Task wurde ein MAP-Befehl abgesetzt. Der Abend-Code ist in der Statistik zu sehen.

'QSFV' Die Directory ist voll.

'QSFZ' Es sind zuviele Felder in einer Map.

'U098' Im CPG2..Druckprogramm enthält die Ausgabe nur X'00'.

'U099' Im CPG2..Druckprogramm ist die Ausgabe größer als 1904 Bytes.

Fehlermeldungen bei der Ausführung 6970

Bei der Ausführung von Online-Programmen unter dem TP-Monitor CICS können folgende Fehlermeldungen auftreten:

CICS-Absturz

Ursache: Storage Violation (siehe Speicherverletzung).

Behebung: 1. Dump ausdrucken. Die Transaction mit mehr als 20 Seiten im Dump ist der Verursacher. Nach dem Restart muss das Programm sofort mit DISABLE abgehängt werden.

Speicherverletzung

Ursache:
1. TWA-Size zu klein (siehe CICS-Absturz). (Wird bei Programmen die ab Release 1.2 umgewandelt wurden, geprüft).
2. Die Operationen BEGAS und ENDAS wurden benutzt und ein Assembler-Statement überträgt Daten zu einer Adresse außerhalb des Programms.
3. Die Operation COPY wurde benutzt und ein Assembler-Statement überträgt Daten zu einer Adresse außerhalb des Programms.
4. Satzlänge in der Dateizuordnung für ein Dataset ist zu klein.
5. Bei Temporary Storage haben zwei Segmente verschiedener Länge den gleichen Namen.
6. Assembler-Fehlermeldungen im CPG-Programm wurden nicht beachtet.
7. Vor einem SYNCP-Befehl wurden Dateien nicht ordnungsgemäß mit RNDOM freigegeben.
8. Bei der Satzlänge für VBOMP wurde der Praefix nicht mitgerechnet.
9. Bei der ADD-Funktion wurde 2 mal die gleiche Datei mit EXCPT angesprochen.
10. In der Ausgabe sind unter einer EXCPT Zeile 2 mal DIAG mit Bezugszahlen verwendet worden.

Behebung: Im CICS-DUMP 'DFHCSA' aufsuchen. Ab 'Storage +1' plus 4C steht die Adresse der letzten aktiven Transaktion.

CICS-Loop

Ursache:
1. Die Druckausgabe im CPG-Programm hat keine positive Ausgabezeile.
2. Zwei EXCPT-Zeilen sind ohne Ausgabefeld hinterein-

ander codiert worden.

3. VSAM-Datei-Fehler (Task wurde beim CA-Split hart abgebrochen.

Behebung: CICS canceln und Dump ausdrucken. Bei Dateifehler muss Forward-Recovery gefahren werden.
Bei Task-Fehler muss nach dem Restart das Programm mit DISABLE sofort abgehängt werden.

CICS Data Base Loop

Ursache: Eine VSAM-Datei mit SHR 3 wurde im Batch bzw. online zerstört.

Behebung: CICS canceln. Im Register 1, in der Partition-SAVE-AREA, steht die Adresse der defekten VSAM-Datei (ACB).
Online-VSAM-Dateien mit SHR 3 ermitteln.

In der FCT haben die Dateien den Eintrag 'ACCMETH= (VSAM, KSDS,KEY)'. Auf die entsprechende Datei Verify fahren.

Bei erfolglosem Verify muss Delete/Define Cluster gefahren werden.

CICS Terminal Dataset Loop

Ursache: Lokale Steuereinheiten waren bei CICS-Start nicht Ready.

Behebung: CICS canceln. IPL für VSE durchführen. Lokale Steuereinheiten einschalten. CICS neu starten.

Transaktionen blockieren

Ursache: Eine Update-Datei ist für andere Tasks gesperrt.

Für ein dialogorientiertes Programm mit DTB = YES in der PCT, ist der Befehl 'SYNCP' falsch bzw. nicht codiert worden.

Behebung: 1. CSMT Term,Outserv,all
2. Dump-Bereich switchen
3. CSMT Term,inserv,all
4. Dump Bereich ausdrucken
Für die blockierten Tasks die entsprechenden Dumps dem Anwendungsprogrammierer übergeben.
Der entsprechende Dateiname befindet sich im Task Dump 'TCA USER' ab Stelle X'84'.

VSAM Illogic Error

Ursache: 1. Pseudo-Satz fehlte bei ADD-Datei.

2. Delete /Define war fehlerhaft.

3. Bei einer VSAM-Datei mit variabler Satzlänge wurden die vier Byte Verschiebung vergessen.

4. Bei einer ESDS/RRDS-Datei wurde nicht die richtige RBA gefunden.

Behebung: Der genaue VSAM-Error-Code wird am Bildschirm angezeigt.

CICS arbeitet ungewöhnlich langsam

Ursache:
1. VSIZE der CICS Partition ist zu klein.
2. In der PCT wurde DTB = YES angegeben und im CPG Programm wurde kein 'SYNCP' gegeben.
3. AMXT in der SIT ist zu klein angelegt.
4. Im CPG-Programm werden auf eine oder mehrere Dateien (VBOMP) sehr viele Zugriffe gestartet.
5. Priorität von CICS ist falsch vergeben worden.

Behebung:
1. VSIZE vergrößern.
2. CICS canceln und Dump ausdrucken. CICS-Dump auf belegte Speicherplätze im DTB-Bereich untersuchen. Nach dem Restart das ermittelte Programm mit DISABLE abhängen.
3. Mit 'CSMT AMX' die Anzahl der aktiven Tasks erhöhen. Anschließend DFHSIT mit neuer AMX umwandeln.
4. CICS-Statistik ausdrucken. Im Abschnitt 'File Statistic' die aufgeführten Dateien überprüfen.
5. Die Priorität für CICS muss vom Operator hochgesetzt werden.

Task Abbruch mit ASRA

Ursache:
1. Falsche Ausgabe-Schablone.
2. Gepackte Daten ohne 'PAC' eingelesen.
3. Ungepackte Daten mit 'PAC' eingelesen.
4. Ein numerisches Feld enthält keine gepackten Daten.
5. Fehler in einer Assembler-Unterroutine.
6. Durch Speicherverletzung ist die Task zerstört worden
7. Ein CPG-Programm wurde mit XCTL aus einem Assembler-Programm aufgerufen, dabei wurde vom Assembler-Programm vor dem XCTL die TWA nicht initialisiert.
8. Wenn bei EXPR und CSA-Verschiebung, so ist die Verschiebung nicht im CPGUCCBA und somit nicht im CPGCCI oder nicht in der CPGURSIT und somit nicht in CPGWRK.

Behebung:
1 bis 5. Dump ausdrucken, fehlerhaftes Feld suchen und Fehlerursache beheben.

6. Dump ausdrucken und Speicherverletzer feststellen. Den Speicherverletzter mit DISAB abhängen. Task mit 'CSMT NEW,PGRMID= ' laden.

7. Vor XCTL ein XC 256(116,12),256(12) legen (bei normaler CPG-TWA), sonst Verschiebung (z. B. bei Datasets) entsprechend berücksichtigen.

Task Abbruch mit AICA

Ursache:
1. Programmloop. Eine GOTO-Operation verzweigt zu einem vorher liegenden TAG und die Schleife wird nicht durch ein READ Bildschirm oder ein WAIT unterbrochen.

2. Im CPG-Programm wurde in der Subroutine auf BEGSR oder ENDSR mit GOTO gesprungen.

3. Eine Rechenroutine wird ohne Ein-/Ausgabe- Unterbrechung so oft durchlaufen, dass die vom CICS festgelegte Maximalzeit erreicht wird.

4. Innerhalb einer DO-Schleife wurde die Operation CLEAR verwendet.

5. Bei einer DO WHILE-Operation wurde versehentlich eine Bezugszahl eingetragen.

Behebung:
1. Schleife entfernen oder unterbrechen.

2. Für GOTO in der SR-Routine muss eine TAG-Zeile definiert werden.

3. Zeit im CICS höher setzen oder Rechenroutine durch ein 'WAIT' unterbrechen.

Task Abbruch mit ASCF

1. In der Eingabe für Temp. Storage wurden Gruppenstufenschalter eingesetzt.

Task Abbruch mit ATNI

1. Es wurde versucht, nicht darstellbare Zeichen auf dem Bildschirm auszugeben.

Task Abbruch mit APCT

1. Ein HL1-Modul wurde nicht gefunden.

Fehlermeldungen bei der Ausführung 6980

CICS- und Batch-Fehlermeldungen sind weitgehend gleich. Folgende Konsolmeldungen sind für Batchverarbeitung unterschiedlich zur CICS-Meldung:

TSQUEUING ERROR

Ursache: Falsche Satznummer bei der Temporary-Storage-Simulation

Behebung:
1. In der Regel ist die maximale Anzahl von Sätzen erreicht (32768).
2. Der Fehler kann auch auftreten, wenn auf eine Satznummer zugegriffen wird, die nicht existiert.

Tabellen 7000

Übersicht der Rechenbestimmungen 7000

Die folgende Tabelle gibt eine Übersicht über die möglichen Eintragungen bei den einzelnen Rechenoperationen. Vergleiche hierzu die Regeln der Grammatik in Abschnitt 3810. Die allgemeine Darstellungsform einer Operation ist dort wie folgt beschrieben:

```
(ON)  OP                                                    (SV)  (BD)
      (F1)  OC  (OE) (DY) (F1) (DY) (OK) (F2) (DY) (EG) (LG)
```

Die Abkürzungen haben die in der folgenden Aufstellung beschriebene Bedeutung. Klammerausdrücke können wahlweise verwendet werden, nicht eingeklammerte Ausdrücke sind zwingend vorgeschrieben.

ON	Abfrage von Bedingungen
OP	Operation
OC	Operations-Code (ADD, READ, DO)
OE	erweiterter Operations-Code (WHILE, UNTIL)
F1	Faktor 1
OK	Operator (größer kleiner gleich)
F2	Faktor 2
EG	Ergebnis
DY	Dummy-Wort (IS FROM TO THAN THEN)
LG	Längenfeld (n u r bei den OCs DLI und QSSA)
SV	Serviceabfrage (mögliche Einträge siehe Kap. 3810)
BD	Bedingungen setzen (Schalter 01 bis 99)
BO	Logische Verknüpfung mit AND und OR

In der folgenden Aufstellung werden die unterschiedlichen Darstellungen für alle Operationen aufgeführt.

```
(ON)     F1    ACCEPT    F2 (KW EG) (SV) (SV) BD
(ON)    (F1)   ADD       F2 EG (SV) (BD)
(ON)           AFOOT     F2 EG (SV) (BD)
(ON)           AVERAGE   F2 EG (SV) (BD)
         F1    BEGDT
         F1    BEGSR
(ON)           BITOF     F2 EG
(ON)           BITON     F2 EG
(ON)           BREAK     (SV)
(ON)           CAB       F1 OK F2 (DY) EG (BD)
(ON)           CALL      F2 (EG)
(ON)     F1    CALLD     F2 (SV) BD
(ON)           CALLM     F2 (EG)
(ON)           CAS(E)    F1 OK F2 (DY) EG (BD)
(ON)           CBS       F1 OK F2 (DY) EG (BD)
(ON)     F1    CHAIN     F2 (KW EG) (SV) (SV) (BD)
(ON)     F1    CHANG(E)  F2
(ON)           CHECK     F2 (SV) (BD)
(ON)           CLEAR
```

```
(ON)           CLEAR-IND EG (SV) BD
(ON)           CLOSE     F2 (SV) (BD)
(ON)           CLRIN     EG (SV) BD
(ON)     F1    COMP(ARE) F2 (KW EG) BD
(ON)           COMRG     EG
(ON)           COM-REG   EG
(ON)           CONT(INUE)
(ON)           CONVERT   (F2 INTO) EG (SV)
(ON)           CONVT     (F2 INTO) EG (SV)
               COPY      F2
(ON)           DEBUG     (SV)
(ON)           DELC      F2 EG
(ON)    (F1)   DELETE    F2
         F1    DEQ(UEUE) (SV)
(ON)     F1    DISPLAY
(ON)    (F1)   DIV       F2 EG (SV) (BD)
(ON)     F1    DLI       F2 EG LG (SV) BD
(ON)           DO        (OE) (DY) (F1) (OK) (DY) (F2) (DY) (EG) (SV)
oder           DO        OE F1 OK F2 (BO)
(ON)     F1    DSPLY
(ON)           DUMP      (F2)
(ON)           EDIT      EG ((TYPE) F2) (SV) (BD)
(ON)           ELIM(INATE) F2 EG
               ELSE
(ON)           END
               END-EVALUATE
(ON)           ENDDO
               ENDDT
               ENDEV
               ENDIF
        (F1)   ENDSR
               ENQ(UEUE)  F2 (SV)
(ON)           EREAD      F2 (SV) (BD)
               EVALUATE
(ON)           EXCPT      (F2) (SV) (BD)
(ON)           EXECUTE    F2 (SV)
(ON)           EXHM       F2 (EG) (SV)
(ON)           EXHM-VAR   F2
(ON)           EXITD      EG (SV)
(ON)           EXITI      F2
(ON)           EXITP      F2 (SV)
(ON)           EXITP-VAR  F2 (SV)
(ON)    (F1)   EXITS      F2 EG (SV)
(ON)           EXITT      F2 (SV)
(ON)           EXITT-VAR  EG
(ON)    (F1)   EXIT-SEND  F2 EG (SV)
(ON)           EXIT-TRANS F2 (SV)
(ON)    (F1)   EXPR       F2 (SV)
(ON)           EXPR-VAR   F2 (SV)
(ON)           EXSR       F2 (BD)
(ON)           FILL       F2 (DY) EG
(ON)     F1    FIND       F2 (BD)
(ON)           GETCHANNEL
(ON)           GETHS
(ON)           GETIN      BD
(ON)           GETMI      BD
(ON)     F1    GET-UPDATE F2 (KW EG) (SV) (SV) BD
(ON)           GO(TO)     F2 (SV)
(ON)           IF         F1 (DY) OK F2 (DY) (BO)
(ON)           IF         SV BD (BD) (BD)
         F1    IF-DAT     F2    (BO)
```

```
          F1    IF-DATI     F2    (BO)
          F1    IF-DATK     F2    (BO)
(ON)            INDOF       (BD)
(ON)            INDON       (BD)
(ON)            JLB         F2
(ON)            JRB         F2
(ON)            JRC         F2 EG
(ON)            JRZ         EG
(ON)            LEFT-SHIFT  F2
(ON)    (F1)    LIST        F2 (DY) (SV) (EG)
(ON)            LIST-VAR    EG
(ON)            LOADT       F2 (SV)
(ON)            LOADT-VAR   F2 (SV)
                LOKUP       F1 OK F2
                LOOK-UP     F1 OK F2
(ON)    (F1)    MACRO       F2
(ON)            MAP         F2 (SV1) (SV2)
(ON)            MAP-VAR     EG (SV)
(ON)            MAPD        F2 (SV1) (SV2)
(ON)            MAPD-VAR    EG (SV)
(ON)            MAPI        F2 (SV1) (SV2)
(ON)            MAPI-VAR    EG (SV)
(ON)            MAPO        EG (SV)
(ON)            MAPO-VAR    EG
(ON)     F1     MAPP        F2 (SV1) (SV2)
(ON)     F1     MAPP-VAR    EG (SV)
(ON)            MLLZO       F2 EG
(ON)            MOVE(R)     F2 (DY) EG (SV)
(ON)            MOVEA       F2 (DY) EG
(ON)            MOVEL       F2 (DY) EG (SV)
(ON)            MOVEN       F2 (DY) EG (SV) (BD)
(ON)            MOVEV       F2 (DY) EG (SV)
(ON)            MOVE-ARRAY  F2 (DY) EG
(ON)            MOVE-LEFT   F2 (DY) EG (SV)
(ON)            MOVE-REST   EG
(ON)            MOVE-RIGHT  EG (SV)
(ON)            MOVE(R)     F2 (DY) EG (SV)
(ON)    (F1)    MULT        F2 (DY) EG (SV) (BD)
(ON)            MVR         EG
(ON)            OPEN        F2 (SV) (BD)
                PARM        F2
                PARAMETER   F2

(ON)            PERFORM     F2 (BD)
                PRNT        F2
                PROGRAM(M)  F2 (EG)
(ON)            PROT        F2 (SV)
(ON)            PROTECTION  F2 (SV)
(ON)            PURGE       F2
(ON)            PUTIN       BD
(ON)            PUTMI       BD
(ON)    (F1)    QSSA        F2 EG LG (SV) OK (SV)
(ON)            RANDOM      F2
(ON)    (F1)    READ        F2 (SV)
(ON)    (F1)    READB       F2
(ON)    (F1)    READB-PAGE  F2 EG
(ON)            READI       F2 (SV)
(ON)    (F1)    READP       F2 EG
(ON)    (F1)    READ-BACK   F2
(ON)    (F1)    READ-PAGE   F2 EG
(ON)            RECEIVE     F2 (SV1) (SV2)
(ON)    (F1)    REPLACE     F2 EG
```

```
(ON)    (F1)   REPLC      F2 EG
(ON)           RIGHT      F2
(ON)           RIGHT-CHAR F2 EG
(ON)           RIGHT-ZERO EG
(ON)           RNDOM      F2
(ON)           ROLL       EG
(ON)           ROLLB      EG
(ON)           ROLL-BACK  EG
(ON)           SAVET      F2 (SV)
(ON)           SAVET-VAR  F2
         F1    SCAN       F2 (EG) (BD)
(ON)           SCREENDUMP (F2)
(ON)           SDUMP      (F2)
(ON)           SELCT      EG (TYPE F2) (SV)
(ON)           SELECT     EG (TYPE F2) (SV)
(ON)           SEND       EG (SV)
(ON)           SETIN      EG (SV) BD
(ON)           SETIX      (F2) EG (SV) BD
(ON)     F1    SETLL      F2
(ON)           SETOF      BD
(ON)           SET(ON)    BD
(ON)           SET-INDEX  (F2) EG (SV) BD
(ON)           SET-INDIC  EG (SV) BD
(ON)     F1    SET-LIMIT  F2
(ON)           SORT(A)    F2 (SV)
(ON)           SQRT       F2 EG (SV)
(ON)    (F1)   SUB        F2 (DY) EG (SV) (BD)
(ON)           SYNCP(OINT) (SV)
         F1    (TAG)
(ON)           TESTB      F2 EG BD
(ON)           TESTF      F2 (DY) EG (SV)
(ON)           TESTN      EG (SV) BD
(ON)           TESTT      F2 BD
(ON)           TEST-BIT   F2 EG BD
(ON)           TEST-FIELD F2 (DY) EG (SV)
(ON)           TEST-NUM   EG (SV) BD
(ON)           TEST-TERM  F2 BD
(ON)           TIME       EG
(ON)           TWALD      F2 (SV)
(ON)           TWASV      F2 (SV)
(ON)           TWA-LOAD   F2
(ON)           TWA-SAVE   F2
(ON)           TWALD-VAR  F2
(ON)           TWASV-VAR  F2
(ON)           UCTRN      F2
(ON)     F1    UPDAT(E)   F2 EG
(ON)     F1    USSA       (SV)
(ON)     F1    VBOMP      F2 EG
(ON)           VSLCT      F2
(ON)           WAIT       (F2)
               WHEN       F1 OK F2 (BO)
(ON)     F1    WRITE      F2 EG
(ON)           XFOOT      F2 EG (SV) (BD)
(ON)           Z-ADD      F2 (DY) EG (SV) (BD)
(ON)           Z-SUB      F2 (DY) EG (SV) (BD)

(ON)     EG    =          F1 (OK) (F2) (SV)
```

Übersicht der Dummyworte 7010

Folgende Wörter werden zur Zeit als Dummyworte erkannt und dürfen deshalb nicht als Feldnamen verwendet werden:

BE	BEI	BY	
FROM			
GIVING			
INTO	IS		
MIT			
ON			
SHALL			
THAN	THEN	TIMES	TO
WITH			

Reservierte Namen 7015

Folgende Namen sind reserviert und dürfen nicht als Feldnamen benutzt werden:

- Operationscodes

- Dummyworte

- CPG-interne Feldnamen
  - CPGxxx
  - Datums- und Zeitfelder wie UDATE, UTIME etc.
  - CP0000 - CP9999, die intern für Namen mit mehr als sechs Stellen und interne Tags generiert werden können.
  - CP00 - CP99, die intern für lange Feldgruppennamen generiert werden können.

Darüberhinaus ist es empfehlenswert, folgende reservierte Namen zu beachten:

Phasen: Mit den verschiedenen CPG-Ausbaustufen wird eine große Anzahl von Phasen ausgeliefert. Diese beginnen mit CPG....
HL1
HMH
HMQ

Masken: Die im Rahmen des CPG ausgelieferten QSF-Masken haben Namen, die mit Q beginnen.

Storages: Die in mitgelieferten CPG-Programmen verwendeten Storages haben Namen, die mit TP oder Q beginnen.

CPG-Bildschirmattribute 7020

Die folgende Tabelle zeigt die Feldeigenschaften in Abhängigkeit von den Aufbereitungsschlüsseln.

CPG Attribut	CICS Hex	Geschützt	Anzeiger D:Doppelhell :Normal hell	Nur numerisch	Lichtstift	MDT gesetzt	Farbe Ter	Farbe Dru
U	40	N					G	
K	C1	N				J		Sch
Q	C4	N			J		r	
B	C5	N			J	J		hwa
							u	
W	50	N		J				rz
E	D1	N		J		J	e	
J	D4	N		J	J			
G	D5	N		J	J	J	n	
A	C8	N	D		J			
C	C9	N	D		J	J	R	
N	D8	N	D	J	J		o	
H	D9	N	D	J	J	J	t	
X	4C	N	Keine					
D	4D	N	Keine			J		
Y	5C	N	Keine	J				
Z	5D	N	Keine	J		J		
0	60	J						
1	61	J				J	B	
2	E4	J			J			
3	E5	J			J	J	l	
S	F0	J					a	
R	F1	J				J		
I	F4	J			J		u	
O	F5	J			J	J		
4	E8	J	D		J		W	Gru
5	E9	J	D		J	J	e	r
P	F8	J	D		J		i	u
L	F8	J	D		J		s	e
M	F9	J	D		J	J	s	n
6	6C	J	Keine					
7	6D	J	Keine			J		
8	7C	J	Keine					
T	7D	J	Keine			J		

Ist kein Attribut angegeben, wird als Default S angenommen.

CPG-Ausgabe Write-Control-Character 7030

Folgende Tabelle zeigt die WCC (Write Control Character) Eintragungen, die bei einer Bildschirmausgabe alternativ zu einer Kombination der Schlüsselworte 'BEEp', 'MODified' und 'LOCked' in der Satzbestimmung der Output Division angegeben werden können.

CPG Eintragung	WCC Hex	Hupe	Modified Data Tag ON	Tastatur verriegelt
K	C0		MOD	LOCK
L	C1			LOCK
M	C2		MOD	
	C3			
N	C4	BEEP	MOD	LOCK
O	C5	BEEP		LOCK
S	C6	BEEP	MOD	
H	C7	BEEP		

Maximalwerte bei einem CPG-Programm 7040

Standardmäßige maximale Programmgröße	24 K
Maximale Programmgröße Batch und HL1-Bausteine	20 K
HL1-Batch standardmäßige maximale TWA-Größe	20 K
Standardmäßige maximale Größe der TWA	4 K
Standardmäßige maximale Länge eines Eingabe- oder Ausgabesatzes	8 K
Maximale Größe der TIOA (Terminal I/O-Area)	4080 Bytes
Maximale Länge alphanumerischer Felder	256 Bytes
Maximale Länge numerischer Felder (15 Ziffern)	8 Bytes
Anzahl Eintragungen in der Namentabelle	2000
maximale Anzahl Feldgruppen	100
maximale Anzahl Dateien und LIST Operationen	100
maximale Anzahl Eintragungen in Datenstrukturen	1000
maximale Anzahl Strukturen	50
maximale Anz. Dateien in der Standard File Tabelle	100
maximale Anz. DO oder IF Operationen im Programm	999
maximale Anz. Subroutinen (BEGSR) im Programm	999
max. Verschachtelung von DO oder IF Operationen	40
maximale Länge von Konstanten (inklusiv Hochkommata für alphanumerische)	
- numerische in der Procedure Division (F1, F2)	10 Stellen
- alphanumerische in Procedure und Output Division	26 Stellen

Eine beliebige Programmgröße mit einer maximalen TWA-Größe von 8K kann alternativ angewendet werden. Das wird durch die Verwendung eines anderen Adressblocks erreicht. Dazu sind die Eintragungen 'BIG' und 'ADD x' in der Options-Parameterleiste vorzunehmen und die resultierenden Einschränkungen zu beachten.

Das Gleiche gilt für eine Vergrößerung der TWA auf 12 K mit dem Options-Parameter 12K. Die notwendige Änderung der Adressierung wird in diesem Fall intern vorgenommen.

Die max. Eingabesatzlänge von 8K kann durch das Feld CPGFIS erweitert werden.

Zwischenspeicher von Daten für CICS-Anwender 7050

1. C W A Common Work Area 7050

Maximale Länge	:	3584
Speichermedium	:	Hauptspeicher
Gültigkeit	:	Systemweit
Definition	:	Systeminitialisierung
Update ?	:	Ja
Adressierung	:	EDIT/SELCT CPGCSA
Eignung	:	Für allgemeine Informationen, welche systemweit zur Verfügung stehen müssen.
Wichtig	:	Es sollten Belegungskonventionen angelegt werden. Die Daten werden nicht automatisch gelöscht.

2. T C T User Area 7055

Maximale Länge	:	255 Bytes, von CPG-Programm 245 Bytes.
Speichermedium	:	Hauptspeicher
Gültigkeit	:	Der Datenstation zugeordnet
Definition	:	In der TCT
Update ?	:	Ja
Adressierung	:	EDIT/SELCT CPGTCT
Eignung	:	Für Anwendungsinformationen, die terminalbezogen zwischengespeichert werden müssen und so auch von einer Task zu einer zu einem beliebigen Zeitpunkt später laufenden anderen Task übergeben werden können.
Wichtig	:	Daten werden nicht automatisch gelöscht.

3. Temporary Storage 7060

Maximale Länge : 8 K

Speichermedium	: Hauptspeicher oder 'Auxiliary' Subpool 5 oder VSAM ESDS
Gültigkeit	: Abhängig von Namenskonventionen
Definition	: Vom ersten Benutzer, der Daten unter einem Namen speichert.
Update ?	: Ja
Eignung	: Zwischenzuspeichernde Informationen
Wichtig	: Ein bestehender TS-Satz darf bei einer Änderung nicht verlängert werden. Aus diesem Grund sollte ein Verzeichnis der einzelnen TS-Queues geführt werden.

Aufbereitungsschlüssel (Edit-Codes) für numerische Felder 7070

Aufbereitungs-schlüssel	Positive Zahl mit 2 Dezimal-stellen	Positive Zahl ohne Dezimal-stellen	Negative Zahl mit 3 Dezimal-stellen
Nicht aufbereitet	1234567	1234567	00120
1	12.345,67	1.234.567	0,120
2	12.345,67	1.234.567	120
3	12345,67	1234567	0,120
4	12345,67	1234567	120
A	12.345,67	1.234.567	0,120CR
B	12.345,67	1.234.567	120CR
C	12345,67	1234567	0,120CR
D	12345,67	1234567	120CR
J	12.345,67	1.234.567	0,120-
K	12.345,67	1.234.567	120-
L	12345,67	1234567	0,120-
M	12345,67	1234567	120-
Y			0/01/20
Z	1234567	1234567	120

Fortsetzung der Aufbereitungsschlüssel siehe folgende Seite.

Aufbereitungs-schlüssel	Negative Zahl ohne Dezimal-stellen	Nullbetrag mit 2 Dezimal-stellen	Nullbetrag ohne Dezimal-stellen
Nicht aufbereitet	00012	000000	000000
1	120	0,00	0
2	120		
3	120	0,00	0
4	120		
A	120CR	0,00	0
B	120CR		
C	120CR	0,00	0
D	120CR		
J	120-	0,00	0
K	120-		
L	120-	0,00	0
M	120-		
Y	0.01.20	0.00.00	0.00.00
Z	120		

Bei der amerikanischen Schreibweise sind Dezimalzeichen und Tausenderpunkt vertauscht und beim Datum wird / statt . verwendet. Um diese Darstellung zu erreichen, muss in der System Initialisation Tabelle CPGURSIT vom Systemprogrammierer ein entsprechender Eintrag vorgenommen werden.

Bei der Datumsaufbereitung mit Y können die führenden Nullen unterdrückt werden. Ist ein Datumsfeld gleich Null, so wird Blank angezeigt. Diese Aufbereitung erreicht man, indem man zusätzlich zum Edit Code Y einen Schutzstern angibt.

Abschnitt 72 Begriffe 7200

Begriffsbestimmungen 7200

Dummywort Ein Dummywort ist ein verbindendes Wort im Text, das den Text der Umgangssprache angleicht, für das Programm jedoch ohne Bedeutung ist. Beispiel: Der Programmtext
IF OTTO IS LESS THAN HUGO kann auch in der Form
IF OTTO LESS HUGO geschrieben werden.
Die Worte IS und THAN sind Dummyworte. Dummyworte können je nach Division oder Benutzeroberfläche unterschiedlich sein.

Feld Ein Feld ist eine bestimmte Anzahl von Speicherstellen zur Aufnahme von Daten, die zu einer Einheit zusammengefasst werden. Ein Feld muss vor Beginn der Verarbeitung definiert werden, das heisst, dem System muss mitgeteilt werden, unter welchem Namen die Daten angesprochen werden welcher Art die Daten sind und wieviele Zeichen das Feld maximal aufnehmen soll. Siehe auch Abschnitt 2, Datenfelder.

Feldgruppe Eine Feldgruppe ist eine bestimmte Anzahl von Feldern mit gleichen Feldeigenschaften (Name, Typ, Länge, Dezimalstellen). Eine Feldgruppe muss wie ein Feld vor der Verarbeitung definiert werden. Siehe auch Abschnitt 2, Feldgruppen. Die einzelnen Felder der Feldgruppe werden auch Elemente genannt. Eine Feldgruppe kann sowohl gruppen- als auch elementweise verarbeitet werden.

Karte Soweit der Begriff Karte im Text dieses Handbuchs verwendet wird, ist damit eine Textzeile oder ein CPG2-Statement gemeint.

Satz Ein Satz ist eine Folge von Worten, die nach den Regeln der Grammatik aneinandergefügt werden und durch ein Satzendezeichen abgeschlossen werden.

Schlüsselwort. Ein Schlüsselwort ist ein Wort, das immer eine Anweisung für den Compiler beinhaltet. Beispiel: Im Text
IF OTTO IS LESS THAN HUGO sind die Worte IF und LESS Schlüsselworte. Schlüsselworte dürfen nicht als Namen für Variable verwendet werden. Schlüsselworte können je nach Division oder Benutzeroberfläche unterschiedlich sein.

Statement Ein Statement ist eine Folge von Worten, die das Programm veranlassen, eine Aktion durchzuführen.

Wort

Ein Wort in Sinne von CPG ist eine Folge beliebiger Zeichen im Programmtext, die einem Blank folgt und durch ein Blank oder Satzende-Zeichen abgeschlossen wird. Ein Wort darf maximal 30 Zeichen lang sein.

Beispiele für Worte:
```
HUGO
 1
 ENDE-DER-FAHNENSTANGE
```

Sonderformen für Worte sind 'Schlüsselworte' und Dummyworte'.

```
HUGO
1
ENDE-DER-FAHNENSTANGE
```

Regeln der Grammatik 7300

Dieser Abschnitt enthält die Regeln der Grammatik in Kurzform. Ausführliche Beschreibung siehe Abschnitt 3.

OPTIONS

```
KW (KW) (DY) (EN) (DY) (KW) (DY) (EN) (DY) ...
```

KW	Schlüsselwort. Als erstes Schlüsselwort ist 'OPTIONS' zwingend vorgeschrieben. Danach können Schlüsselworte nach Abschnitt 3300 in beliebiger Reihenfolge folgen und gegebenenfalls durch Eintragungen oder Dummy-Worte ergaenzt werden.
DY	Dummy-Wort
EN	Eintragung zu einem Schlüsselwort

```
Beispiel - OPTIONS Z/OS LOWER CASE LETTERS TITLE IS BEISPIEL.
             KW      KW  KW    DY   DY      KW    DY EN
```

FILES

```
1.      KW DN
2.      KW DN DN (DN) (DN) ...
3.      KW DN EN (EN) EN (EN) EN (EN EN EN) EN
```

KW	Schlüsselwort. Als Schlüsselwort ist 'FILE' oder oder bei Mehrfachdefinition (Fall 2.) 'FILES' vorgeschrieben.
DN	Dateiname
EN	Eintragung bei manueller Beschreibung (siehe Abschnitt 3400).

```
Beispiel - FILE KUNDEN.
         - FILES KUNDEN ARTIKEL.
         - FILE TEST INPUT FIX 1000 200 5 KEY INDEX DISK
```

DATA DIVISION

```
         FN (EL OK) LG (DP)
oder     KW ST (DY) (SA)
oder     KW (FN)
```

	FN	Name des Feldes oder der Feldgruppe
	EL	Anzahl Elemente einer Feldgruppe
	OK	Operator '*' multipliziert Länge mit Anzahl Elemente
	LG	Länge des Feldes in Zeichen (Stellen)
	DP	Anzahl Dezimalstellen

oder	KW	'DEFINE' für eine Data Dictionary Struktur
	ST	Strukturname
	DY	Dummywort 'TYPE'
	SA	Satzart der Struktur

oder	KW	'ORG' zum Positionieren in der TWA
	FN	Feldname

```
Beispiel - WERT 11 2.            * WERT ist num. 11 Stellen 2 Dez.St.
         - PAGE 24 * 80.         * Feldgruppe 24 Felder je 80 Stellen
         - TEXT 25.              * Alphafeld 25 Stellen
oder     - DEFINE CPGWRK.        * Anlegen der Struktur 'CPGWRK
oder     - ORG.                  * Positionieren auf dem TWA-Anfang
```

FORMS

```
         KW FN (KW LG) ((KW) LN (KW) CH) ((KW) LN (KW) CH) ...
```

	KW	Schlüsselwort: Als erstes Schlüsselwort ist 'FORMS' zwingend vorgeschrieben. 'LENGTH' kennzeichnet die folgende Eintragung als Formularlänge in Zeilen. 'LINE' oder 'ZEILE' kennzeichnet die folgende Eintragung als Zeilennummer. 'CHAN' oder 'KANAL' kennzeichnet die folgende Eintragung als Vorschub-Kanal
	FN	Name des Druckers in der FILES DIVISION
	LG	Länge des Formulars in Zeilen
	LN	Zeilen-Nummer
	CH	Kanal

```
Beispiel - FORMS DR01.
         - FORMS DR02 LENGTH 36
         - FORMS DR03 LINE 3 CHAN 01 LINE 66 CHAN 12.
         - FORMS DR05 LENGTH 36 003 01 066 12.
         - FORMS DR06 LENGTH 36 ZEILE 3 KANAL 1 ZEILE 66 KANAL 12.
```

INPUT DIVISION

Satzbestimmung

```
KW DN (VAR) (BA)
```

KW	Schlüsselwort 'FILE' oder 'FIELD'
DN	Name der Datei (bis zu acht Stellen lang)
VAR	variable Eingabepositionen bei großen Dateien
BA	Bedingungsabfrage (siehe unten)

Bedingungsabfrage (BA)

Die Bedingungsabfrage ist für die einzelnen Dateitypen unterschiedlich. Die allgemeine Form richtet sich daher nach dem Dateityp:

Platte	(TP) (BD) (CA) (AND) (CA) (AND) (CA)
CA =	PS (NOT) CD CH
Dataset	(TP) (BD) (CA) (AND) (CA) (AND) (CA)
Bildschirm	(TP) (BD)
Data Dictionary	TP (KW SA)
Datenstruktur	TP (LG)
HL1 Datenkanal	TP

In dieser Darstellung bedeuten:

TP	Dateityp (z.B. DS = Datenstruktur)
BD	Bedingung (Schalter 01 bis 99)
SA	Satzart (über Schlüsselwort 'TYPE')
CA	Zeichen Abfrage
PS	Position im Satz
CD	Zeichen Code (Character Digit Zone)
CH	Zeichen
LG	Länge der Datenstruktur
AND	Verknüpfung mehrerer Abfragen
NOT	Umkehrung (Bedingung ist nicht erfüllt)

Beispiel

```
- FILE BILD.
- FILE PLATTE.
- FILE KUNDEN DD.
- FILE PLATTE KF 11 1 NOT CHAR A 2 CHAR B 3 CHAR #.
- FILE ARTIKEL AA 01 1 C A AND 2 C R.
- FILE OTTO DS.
- FILE HUGO DS 200.
- FILE MYROOT DS.
- FILE XKANAL HS.
- FILE A0001 HS.

- FIELD HUGO.
- FIELD ALPHANUMERIC-STRING.
```

Feldbestimmung

```
          (SF) VP BP (DP) FN (SP) (BD)
```

SF = Speicherungsform (packed, binary, logical)
VP = von Position im Satz (num. Wert max. 4 Stellen)
BP = bis Position im Satz (num. Wert max. 4 Stellen)
DP = Anzahl Dezimalstellen (Ziffer von 0 bis 9)
FN = Feldname
SP = Schlüsselwort für Lichtstiftbezugszahl
BD = eine Lichtstiftbezugszahl oder ein Gruppenstufenschalter L0 bis L9 oder bis zu drei Eingabebezugszahlen

Beispiel

```
- 1 2 SA.                    * 1 bis 2 Alphafeld SA
- 3 9 2 WERT.                * 3 bis 9 WERT mit 2 Dezimalen
- P 10 11 0 LNR.             * 10 - 11 LNR  gepackt
- 1 3 0 DIV # # 99.          * Schalter 99, wenn DIV = 0
- 511 520 KNR  SP 01.        * Schalter 01, wenn KNR gewählt
-  11 13  WGRP  L1.          * Gruppenwechsel L1 bei WGRP
```

PROCEDURE DIVISION

```
(ON) OP (SV) (BD)
```

ON	Bedingungsabfrage
OP	Ausführung der Operation
SV	Serviceabfrage
BD	Bedingungen setzen

Die Bedingungsabfrage (ON).

```
KW (NOT) BD (AND NOT) BD (AND NOT) BD
```

KW	Schlüsselwort 'ON'
BD	Bedingungsschalter (01 BIS 99, Px, Tx, Ax, Cx, DE..)
AND	Dummywort Und-Verknüpfung
NOT	Umkehrung (Bedingung nicht erfüllt)

```
OC (OE) (DY) (F1) (DY) (NOT) (OK) (F2) (DY) (EG)
kurz:  OC (F1) (F2) (EG)
```

OP	Operations-Schlüsselwort (siehe Abschnitt 4)
OE	erweiterter OP.-Code (UNTIL, WHILE ...)
F1	Faktor 1 (Feldname)
F2	Faktor 2 (Feldname)
EG	Ergebnis (Feldname)
DY	Dummy-Worte (FROM, TO, WITH, TIMES ...)
OK	OPERATOREN (GT, LT, EQ, >, <, =)

Beispiele:

- EXCPT	OP
- DO 10 TIMES	OP F2 DY
- DO FROM X TO Y	OP DY F1 DY F2
- MOVE TEXT TO LINE	OP F1 DY F2
- X = 0	EG OP F2
- X = A + B	EG OP F1 OK F2
- READ OTTO	OP F2
- MAP ARTIKEL	OP F2
- OBEN	F1
- OBEN TAG	F1 OP

OUTPUT DIVISION

Satzbestimmung

```
KW DN (KW) (ON) (LB)
```

KW	1. Schlüsselwort 'FILE' oder 'FIELD'
DN	Dateiname (max 8 Stellen) oder Feldname
KW	Schlüsselworte je nach Einheit Bildschirm (ERASE, UNP, BEEP, MOD, LOCK) Platte (ADD, DEL, ALG) Drucker (SPACE X X, SKIP)
ON	Bedingungabfrage (wie Procedure Division)
LB	Anspringpunkt (Merkmal)

Feldbestimmung

```
(ON) FN PS (LT) (AT) (EC) (CR) (CL) (EH)
```

ON	Bedingungsabfrage (siehe Procedure Division)
FN	Feldname
PS	Position im Satz (num. Wert max. 4 Stellen) beim Bildschirm ZZSS (ZZ = Zeile, SS = Spalte)
LT	Literal (Textkonstante oder Schablone)
AT	Bildschirmattribut
EC	Aufbereitungsschlüssel für numerische Werte
CR	CURSOR oder löschen nach Ausgabe (BLANK)
CL	Farbe (W, R, B, G, Y, P, T)
EH	EH Wert (BLINK, REVERSE, UNDERSCORE)

Schlüsselworte in der Procedure Division 7410

Die folgenden Schlüsselworte sind für die Operationen in der Procedure Division reserviert. Sie dürfen nicht als Namen von Variablen oder als Namen von Labeln benutzt werden:

ADD	AFOOT							
BEGAS	BEGDT	BEGSR	BITOF	BITON	BREAK			
CAB	CALL	CALLD	CALLM	CAS	CBS	CHAIN	CHECK	CHANG
	CLEAR	CLOSE	CLRIN	COMP	COMRG	CONT	CONVT	COPY
DEBUG	DELC	DELET	DEQ	DIV	DLI	DO	DSPLY	DUMP
EDIT	ELIM	ELSE	END	ENDAS	ENDDO	ENDDT	ENDEV	ENDIF
	ENDSR	ENQ	EREAD	EXCPT	EXHM	EXITD	EXITI	EXITP
	EXITS	EXITT	EXPR	EXSR				
FILL	FIND							
GETHS	GETIN	GETMI	GOTO					
IF	INDOF	INDON						
JLB	JRB	JRC	JRZ					
LIST	LOADT	LOKUP						
MACRO	MAP	MAPD	MAPI	MAPO	MAPP	MLLZO	MOVE	MOVEA
	MOVEL	MOVEN	MOVEV	MULT	MVR			
OPEN								
PARM	PRNT	PROT	PURGE	PUTIN	PUTMI			
QSSA								
READ	READB	READI	READP	REPLC	RNDOM	ROLL	ROLLB	
SAVET	SCAN	SDUMP	SELCT	SETOF	SETIN	SETIX	SETON	SETLL
	SORTA	SORTA	SQRT	SUB	SYNCP			
TAG	TDUMP	TESTB	TESTF	TESTN	TESTT	TIME	TWALD	
	TWASV							
UCTRN	UPDAT	USSA						
WAIT	WHEN	WRITE						
XFOOT								
Z-ADD	Z-SUB	=						

ACCEPT	AVERAGE			
BIT-OF	BIT-ON			
CHANGE	CLEAR-IND	COMPARE	COM-REG	CONVERT
DEQUEUE	DELETE	DISPLAY		
ELIMINATE	END-EVALUATE	ENQUEUE	EVALUATE	EXECUTE
	EXHM-VAR	EXIT-SEND	EXIT-TRANS	EXITP-VAR
	EXITT-VAR	EXPR-VAR		
GET-UPDATE	GETCHANNEL	GO		
LEFT-SHIFT	LIST-VAR	LOADT-VAR	LOOK-UP	
MAP-VAR	MAPD-VAR	MAPI-VAR	MAPO-VAR	MAPP-VAR
MOVER	MOVE-ARRAY	MOVE-LEFT	MOVE-REST	MOVE-RIGHT
PARAMETER	PERFORM	PROGRAM	PROGRAMM	PROTECTION
RANDOM	READ-BACK	READ-PAGE	READB-PAGE	RECEIVE
	REPLACE	RIGHT	RIGHT-CHAR	RIGHT-ZERO
	ROLL-BACK			
SAVET-VAR	SCREENDUMP	SELECT	SEND	SET
	SET-INDEX	SET-INDIC	SET-LIMIT	SQARE-ROOT
	SYNCPOINT			
TDUMP	TEST-BIT	TEST-FIELD	TEST-NUM	TEST-TERM
	TWA-LOAD	TWA-SAVE	TWALD-VAR	TWASV-VAR
UPDATE				

Online-Programm zum Löschen von Temporary-Storage-Bereichen 7510

Programmgesteuert können Gruppen von temporären Speicherbereichen mit dem HL1-Modul HQDELS gelöscht werden. Dazu wird über einen Datenkanal mit dem Modul kommuniziert.

```
-i
   file hqdels hs.
     1  2  oc.              * Operationscode
     3  4  rc.              * Returncode
     5  8  p1bis4.          * Position 1 bis Position 4 des TS-Namens

-c.
                            * alle TS löschen
   oc = 'L '.               * Operationscode 'Löschen'
   p1bis4 = 'TEST'
   exhm hqdels hqdels t.    * Alle Storages, deren Name mit TEST
                            * beginnt, werden gelöscht
```

Freies Format in starre Schreibweise ändern 7520

Das folgende Programm bietet Ihnen die Möglichkeit, in freier Schreibweise codierte Programme in RPG-ähnlichen Quellencode umzuwandeln.

```
- OPTIONS PHASE xxxxxx
-         BATCH
-         TITEL Freies#Format#stanzen
-         END
- FILE IJSYS03 INPUT  FIX  1600 80  DISK
- FILE OKARTE  OUTPUT VAR       80  PUNCHER
- -I. FILE IJSYS03.
-        1 80 SATZ
-        6  7 CHECK
- -C.
-     DO UNTIL CPGFRC >< '  '.    * Bis End of File
-        READ IJSYS03
-        IF CPGFRC = '  ' AND
-        IF CHECK >< '-*'
-           EXCPT
-        ENDIF
-     ENDDO
- -O.
-     FILE OKARTE
-        SATZ   80
```

Vorgehensweise:

Das obenstehende Programm wird eingegeben und umgewandelt. Anschliessend werden in den JCL-Karten des CPG2-Quellenprogramms die ASSGN- und EXEC LNKEDT-Befehle durch den Befehl

// EXEC xxxxxx,SIZE=AUTO (Phasenname des obenstehenden Programms)
ersetzt.

Nach Umwandlung dieses Programms steht das CPG2-Programm in spaltengebundenem CPG-Code in der Punch Queue zur Verfügung.

Die Umwandlung dieses Batchprogramms ist nur mit CPG3 möglich.

Anlegen einer Datenview 7530

In relationalen Datenbanken wird eine Datenview online erstellt und bleibt für die Zeit ihrer Verarbeitung im Hauptspeicher.

Um diese Verarbeitungsform zu simulieren, wird eine View per Programm, CPG3..Query-Programm oder mittels QTS erstellt und auf einer Datei abgestellt. Die View wird erst dann von dieser Datei in den Hauptspeicher geladen, wenn sie in einem Programm mit dem Befehl FIND angesprochen wird. Die geladene View bleibt dann bis zum Shut Down des TP-Monitors im Hauptspeicher. Eine View kann auch in einem Batchprogramm verarbeitet werden, sofern der verfügbare GETVIS-Bereich der Partition für die Verarbeitung ausreicht.

Vorgehensweise für CPG2-Anwender 7531

Die View wird in diesem Fall in zwei Schritten angelegt: Zunächst wird die Struktur der Tabelle mit einem CPG-Programm beschrieben und ihre Elemente mit Werten gefüllt, dann wird mit dem Serviceprogramm CPGZCTB die View dokumentiert und auf die Datei CPGWKV abgestellt.

1. Schritt: Füllen der View mit einem CPG-Programm

Das folgende CPG-Programm zeigt beispielhaft das Füllen einer View. Vereinfachend wird hier unterstellt, dass die gesamte Tabelle über den Bildschirm (über die Map EINGABE) eingegeben wird.

```
- FILE CPGWKV UPD VAR  4020 20  KSDS
- -D.
-      SATZ  0 * 22
-      KDNR       5 0
-      FIRMA     13
-      PLZ        6
-      LNR        5 0
-      KEY       20
- -C.
-      DO LOOP
-         MAPD EINGABE
-         LNR = LNR + 1
-         EDIT  KEY
-         KEY CHAIN CPGWKV CHECK
-         IF CPGFRC = '  '
-            EXCPT ÄNDERN
-         ELSE
-            EXCPT NEU-ANLEGEN
-         ENDIF
-      ENDDO
- -O.
-     FILE CPGWKV  ADD  NEU-ANLEGEN
-        KEY  20
-        SATZ 42
-     FILE CPGWKV       ÄNDERN
-        SATZ 42
-     FIELD KEY
-              2 '22'
-        LNR  20  PACKED
```

Das erstellte Programm dient als Input für das in Schritt 2 beschriebene Serviceprogramm. Deshalb müssen der Dateizugriff und die Schlüsselpositionen unbedingt mit dem Beispiel übereinstimmen.

Das Beispiel ist gewählt worden, um die wesentlichen Karten übersichtlich darzustellen. In der Regel wird in der Praxis ein Batchprogramm zu schreiben sein, das die Daten der View aus einer Datei oder mehreren Dateien zusammenstellt.

2.Schritt: Dokumentieren und Abstellen der View

Zum Dokumentieren und Abstellen der View steht die Phase CPGZCTB zur Verfügung. Es ist ein Job wie folgt zu erstellen:

```
:
// JOB JCPGZCTB
// PAUSE          Datei CPGWKV closen
// EXEC CPGZCTB,SIZE=AUTO
Vorlaufkarte
/*
// PAUSE          Datei CPGWKV eröffnen
/&
```

Die Vorlaufkarte hat folgenden Aufbau:

Stellen 1 - 4	Name der Tabelle, mit dem sie im Befehl FIND angesprochen wird.
Stellen 5 - 7	Länge der Tabellenelemente
Stelle 8	L für 'Löschen einer Tabelle' (optional)
Stellen 10 - 12	Personenkurzzeichen des Sachbearbeiters (optional)
Stellen 13 - 40	Dokumentation, Teil 1 (optional)
Stellen 41 - 80	Dokumentation, Teil 2 (optional)

Vorgehensweise für CPG3-Anwender 7532

CPG3-Anwendern wird das Handling von Datenviews erleichtert. Zum Erstellen einer View kann z.B CPG3..Query benutzt werden. Das Serviceprogramm QTS (Quick Table Service) ermöglicht online das Löschen einer View aus der Datei oder im Hauptspeicher, das Laden einer neuen Version aus der Datei und das Anzeigen aller angelegten Tabellen.

Diese Serviceprogramme sind im CPG3-Handbuch beschrieben.

CPG-Umwandlung ohne IJSYS04 7540

Mit folgendem Job wird ein Beispiel aufgezeigt, wie CPG-Umwandlungen ohne ASSGN-SYSIN-Anweisung ermöglicht werden. Bei dieser Ausführung wird IJSYS04 nicht benötigt.

Diese Art der CPG-Umwandlung bringt Laufzeitverbesserungen, wenn 3380 Platten installiert sind oder wenn die Umwandlungsbereiche über VSAM Space Manager verwaltet werden.

```
* $$ JOB CPGUMW
* $$ PUN DISP=I
// JOB CPGUMW
// EXEC CPGZPUN,SIZE=AUTO
       * $$ JOB CPGASS
       // JOB ASSEMBLY
/*
// EXEC CPG2
     - OPTIONS NOSYSIN.
     :
     :    CPG PROGRAMM
     :
/*
// EXEC CPGZPUN,SIZE=AUTO
      // EXEC LNKEDT
      /&
      * $$ EOJ
/&
* $$ EOJ
```

Erläuterungen zu den JCL-Karten:

Die Power-Punch-Anweisung mit dem Parameter DISP=I stellt die gestanzten Karten in die Power Reader Queue ab.

Als Options-Parameter muss NOS für No SYSIN eingetragen werden. Die Phase CPGZPUN ist im Lieferumfang von CPG enthalten.

CPG für ESA-Command-Level-Programme ohne Methodenbank 7550

Solche Programme, die die Methodenbank nicht nutzen und mit EXEC CICS-Commands arbeiten, erkennt man am Options-Parameter CICSESA oder der Kombination CICS ESA.

Die Umwandlung eines solchen Programms wird in drei Steps durchgeführt. Die jeweilige Eingabe wird über die Punch Queue mit DISP=I weiterverarbeitet.

```
STEP 1 :  CPG-Umwandlung  mit        EXEC CPG / CPG2 / HL1
STEP 2 :  Command Level Prepocessor  EXEC DFHEAP1$
STEP 3 :  Assembly mit Link          EXEC ASSEMBLY
```

```
 * $$ JOB JNM=CPGUMW
 * $$ PUN DISP=I
 // JOB CPGUMW
 // EXEC CPGZPUN,SIZE=AUTO
    * $$ JOB JNM=CPGPREP
    * $$ PUN DISP=I
    // JOB CPGPREP               JOB DFHEAP1$
    // EXEC CPGZPUN,SIZE=AUTO
       * $$ JOB JNM=CPGASM
       // JOB CPGASM             JOB ASSEMBLY
 /*
 // EXEC CPG
      - OPTIONS  NOSYSIN  CICS ESA.
      :
 /*
 // EXEC CPGZPUN,SIZE=AUTO
    // EXEC CPGZPUN
       /*
       // EXEC LNKEDT
       /&
       * $$ EOJ
    /&
    * $$ EOJ
 /&
 * $$ EOJ
```

Erläuterungen zu den JCL-Karten:

Die Power-Punch-Anweisung mit dem Parameter DISP=I stellt die gestanzten Karten in die Power Reader Queue ab.

Als Options-Parameter muss NOS für No SYSIN eingetragen werden. Die Phase CPGZPUN ist im Lieferumfang von CPG enthalten.

SQL/DS 7570

Ab der Ausbaustufe CPG3 ist die Sprache SQL in CPG-Programmen direkt unterstützt. SQL-Statements können so in den CPG-Code eingebunden werden, wie der Hersteller es vorgibt. Zur Kennzeichnung beginnen SQL-Statements mit dem Schlüsselwort SQL.

Ein Beispiel für die SQL-Verarbeitung im CPG3 mit Job Control Statements:

```
// JOB CPGSQL
// LIBDEF PROC,SEARCH=(PRD2.SQL220)
// EXEC PROC=ARISLIBP   *-- SQL/DS PRODUCTION LIBRARY ID PROC
// ON $ABEND GOTO REASS
// EXEC CPGPREP,PARM='USERID=SQLDBA/SQLDBAPW'
     - OPTIONS ROOT PHASE TST026  TITEL SQL-TESTPROGRAMM.
     - -D.
     - SQL  BEGIN DECLARE SECTION
     -         USER           8
     -         PASSW          8
     -         KDNR           5 0
     -         FIRMA         30
     - SQL  END DECLARE SECTION
     - -C.
     -      USER = 'SQLDBA  '
     -      PASSW = 'SQLDBAPW'
     - SQL  CONNECT :USER IDENTIFIED BY :PASSW
     - SQL  DECLARE C2 CURSOR FOR
     - SQL     SELECT KKDNR,KFIRMA
     - SQL     FROM KUNDENA
     - SQL     WHERE KKDNR  < 3000
     - SQL  OPEN C2
     -      DO UNTIL CPGMPF = 'P3'
     -        SQL FETCH C2
     -        SQL   INTO  :KDNR,:FIRMA
     -        MAPD BILD.             * Der Return-Code steht im Feld
     -                               * SQCODE
     -      ENDDO
     -      SQL  CLOSE C2
     -      SQL  COMMIT WORK
     -      MAPO ENDE
/*
// IF $RC NE 0 THEN
// GOTO ENDE
*  STEP HL1
// LIBDEF PHASE,CATALOG=SP4U.ULIBL
// DLBL IJSYSIN,'F4.WORK.04',0,SD,,CISIZE=8192
// EXTENT SYSIPT,PRD201,1,0,46000,4000
ASSGN SYSIPT,122
// EXEC HL1
/*
// IF $RC NE 0 THEN
// GOTO REASS
*  STEP ASSEMBLER
CLOSE SYSIPT,READER
ASSGN SYSIN,122
*  STEP LNKEDT
// EXEC LNKEDT
/*
```

```
// IF $RC EQ 0 THEN
// GOTO ENDE
/. REASS
CLOSE SYSIPT,READER
/. ENDE
/&
```

Beispiel 1: Platten-Änderung. 8005

PLATTEN-ÄNDERUNG KDNR KURZNAME TT.MM.JJ

```
 1 -  OPTIONS TITEL PLATTEN-ÄNDERUNG PHASE TEST.
 2 -  FILE CPGWRK.                  * im Data Dictionary beschrieben
 3 -  DATA DIVISION
 4 -     KEY     14
 5 -  INPUT DIVISION
 6 -     FILE CPGWRK
 7 -        15 100 SATZ
 8 -  PROCEDURE DIVISION
 9 -     DO LOOP
10 -        MAPD EINGABE
11 -        KEY CHAIN CPGWRK
12          IF CPGFRC = '  '.       * Satz gefunden
13 -           MAPD ANZEIGE
14 -           IF CPGMPF = 'P1'
14 -              EXCPT
14 -           END
15 -        END
16 -     ENDDO
17 -  OUTPUT DIVISION
18 -     FILE CPGWRK
19 -        SATZ  100
```

```
Mapname    :  EINGABE
======================

FELD   A F E EC C B ATTFLD ALT.E

KEY    A        C


Mapname    :  ANZEIGE
======================

FELD   A F E EC C B ATTFLD ALT.E

SATZ   A        C
```

Die Statemments wurden links mit einer laufenden Numerierung versehen. Die folgende Erklärung bezieht sich jeweils auf diese Statement-Nummer.

1 Options: Titel ist Platten-Änderung, Phasenname ist TEST.

2 Die Datei CPGWRK wird verarbeitet. Die Dateidaten werden dem Data Dictionary entnommen. (Die Datei kann für Update verarbeitet werden, sie hat eine feste Satzlänge von 100 Bytes, die Schlüssellänge ist 14 Bytes; es handelt sich um eine VSAM-KSDS-Datei.)

3 Hier beginnt die Data Division.

4 Das Feld KEY wird als vierzehnstelliges alphanumerisches Feld für das Programm definiert.

5 Hier beginnt die Input Division.

6 Von der Datei CPGWRK soll gelesen werden (können).

7 Von der Datei CPGWRK werden die Stellen 15 bis 100 in das Feld SATZ eingelesen. Die Eingabestruktur wird in der Regel hier nicht im einzelnen beschrieben. Vielmehr wird sie im Data Dictionary gepflegt und von dort bei der Umwandlung ins Programm eingefügt. Statement 6 wäre dann: FILE CPGWRK DD, die Feldbestimmungen wie hier in Statement 7 würden ganz entfallen.

8 Hier beginnt die Procedure Division.

9 Hier beginnt eine Endlosschleife.

10 Der Bildschirmdialog. Das im QSF beschriebene Bild EINGABE wird hier ausgegeben. Das Programm hält an und erwartet die Eingabe eines Schlüsselfeldes KEY. Bei der nächsten Programmfunktionstaste startet das Programm wieder und liest die modifizierten Felder vom Bildschirm ein.

Mit der 'Lösch'-Taste kann das Programm beendet werden.

11 Mit dem Feld KEY wird auf die Datei CPGWRK direkt zugegriffen. Ist der Schlüssel vorhanden, so wird entsprechend der Input Division von der Datei gelesen.

Ist der Schlüssel nicht vorhanden, wird das CPG-interne Feld CPGFRC (File Return-Code) mit dem Wert 'NF' (not found) gefüllt. Eine Eingabeübertragung findet in diesem Fall nicht statt.

12 Der File Return-Code (siehe Punkt 11) wird hier abgefragt. Die Statements zwischen IF und END werden nur ausgeführt, wenn beim CHAIN ein Satz gefunden wurde.

13 Zweiter Bildschirmdialog. Hier wird das Feld SATZ ungeschützt angezeigt, um den Dateiwert überschreiben zu können.

14 Wurde die Programmfunktionstaste 1 betätigt, so wird in die Output Division verzweigt. Bei jeder anderen Funktionstaste (dazu zählt auch DE - Data Entry) wird dieses Statement nicht ausgeführt.

Die Programmfunktionstaste kann im Feld CPGMPF abgefragt werden. Die Tasten sind auf zwei Stellen komprimiert (im Beispiel P1 für PF1). Siehe dazu Stichwort Programmfunktionstasten (Kapitel 2260)

15 END beendet die IF-Abfrage. Zur besseren Dokumentation kann man auch mit dem Operationscode ENDIF arbeiten.

16 ENDDO beendet die Endlosschleife. Die Ausführung des ENDDO bewirkt, dass zum zugehörigen Schleifenanfang verzweigt wird, also hier zur Zeile 9.

17 Hier beginnt die Output Division.

18 Bei jeder EXCPT-Ausgabe wird auf die Datei CPGWRK ausgegeben.

19 Das Feld SATZ wird auf Stelle 100 der Datei CPGWRK ausgegeben.

Beispiel 2: Dateianzeige mit READ PAGE 8010

Die Datei KUNDEN ist eine indexsequentiell organisierte Plattendatei (VSAM KSDS).

Dateianzeigeprogramm KDNR KURZNAME tt.mm.jj

```
 1      - OPTIONS TITEL Dateianzeigeprogramm ROOT PHASE TEST.
 2      - FILE KUNDEN
 3      - -D
 4      -     PAGE 20 * 79
 5      - -I
 6      -     FILE KUNDEN DD
 7      - -C
 8      -     DO LOOP
 9      -        KDNR READ-PAGE KUNDEN PAGE
10      -        IF CPGFRC = 'EF'.   * File Return-Code 'End of File'
        -           FILL ' ' TO KDNR
        -        END
11      -        RANDOM  KUNDEN
12      -        MAPD ANZEIGE
13      -        FILL ' ' TO PAGE
14      -     ENDDO
15      - -O
16      -     FIELD PAGE
17      -        KDNR     7
18      -        NAME    33
19      -        ORT     55
20      -        UMSATZ  70 EDITCODE K


   Mapname    :  ANZEIGE
   ======================

   FELD   A F E EC C B ATTFLD ALT.E

   PAGE   P
21 ####   A   U    C          KDNR
```

Erläuterungen:

1 CPG2-Steuerkarte.

2 Eine Datei mit dem Namen 'KUNDEN' wird definiert.

3 Beginn der Data Division.

4 Eine Feldgruppe mit dem Namen 'PAGE' wird definiert. Sie besteht aus 20 Feldern von je 79 Bytes Länge und soll eine Bildschirmseite mit 20 Zeilen (Zeilenlänge 79) aufnehmen.

5 Beginn der Input Division.

6 Von der Datei KUNDEN soll ein Satz gelesen werden; die Struktur des Datensatzes wird dem Data Dictionary entnommen. Im Beispiel werden die Input-Feldbestimmungen für die Felder KDNR, NAME, ORT und UMSATZ zur Umwandlungszeit in das Programm eingefügt.

7 Rechenbestimmungen.

8 Beginn einer Endlosschleife.

9 Von der Datei 'KUNDEN' werden so viele Sätze gelesen, wie die im Ergebnisfeld eingetragene Feldgruppe Elemente enthält. Der erste gelesene Satz ist der, dessen Schlüssel gleich oder größer als der Inhalt des Feldes 'KDNR' ist. Die Seite wird in der Feldgruppe 'PAGE' gespeichert. Die Anzahl der Zeilen ergibt sich aus der unter NR.3 definierten Feldanzahl. Zur Aufbereitung der Zeilen siehe Nr. 15 bis 19.

10 Bei Dateiende wird das Feld CPGFRC mit 'EF' wie End of File gefüllt. Es muss unmittelbar nach der READ-Operation abgefragt werden.

Hier wird der Schlüssel für den nächsten Lesezugriff mit Blank gefüllt, um im EF-Fall wieder beim ersten Satz der Datei aufzusetzen.

11 Mit RANDOM wird die Datei KUNDEN freigegeben. Dies ist ein Befehl, der für die Funktion des hier behandelten READ-PAGE nicht erforderlich ist. Vielmehr wird berücksichtigt, dass Anzeigeprogramme in der Regel von mehreren Benutzern gleichzeitig ausgeführt werwerden können. Deshalb wird mit RANDOM vor dem folgenden Bildschirmdialog die Leseoperation beendet, um die VSAM-Strings freizugeben.

12 Die aufbereitete Bildschirmseite wird in der Map ANZEIGE ausgegeben.

13 Der Befehl FILL wird hier auf einer Feldgruppe ausgeführt. Ist die Feldgruppe nicht indiziert, wird FILL für alle Elemente gleich ausgeführt. PAGE wird also auf Blank 'gelöscht'.

14 ENDDO verzweigt zurück zum zugehörigen DO. Für den Programmablauf bedeutet das, dass geblättert wird. Im nächsten Schleifendurchlauf wird wiederum mit 20 Lesezugriffen zur Datei KUNDEN die Feldgruppe PAGE gefüllt. Das Lesen beginnt mit der zuletzt gelesenen KDNR oder mit Blank, falls im letzten Durchlauf End of File erreicht wurde.

15 Hier beginnt die Output Division.

16 Feldaufbereitung. Die Aufbereitung des Feldes PAGE wird durch den Befehl READ-PAGE ausgelöst. Das Feld KDNR wird in Position 1 bis 7 aller Felder der Feldgruppe ausgegeben, das letzte Byte von NAME steht auf Position 33, das letzte Byte von ORT auf Position 55 in jedem Element.

20 Das numerische Feld UMSATZ steht mit seinem letzten Byte auf Stelle 70 in jedem Element der Page. UMSATZ wird mit einem Aufbereitungsschlüssel K (Schlüsselwort EDItcode) aufbereitet: Führende Nullen werden unterdrückt; Dezimalstellen werden durch ein Komma gekennzeichnet; ist das Feld negativ, wird ein Minuszeichen an das Feld angehängt; Tausenderpunkte werden in das Feld eingefügt.

Siehe auch Stichwort Aufbereitung, z.B. Tabelle 7070.

21 In der Maske kann eine 5-stellige Blankkonstante ungeschützt ausgegeben werden. In der Beschreibung gibt man beim 'Zusätzlichen

Eingabefeld' den Feldnamen KDNR ein. Somit kann man in jeder Maske einen Schlüssel angeben, bei dem dann die nächste seitenweise Anzeige beginnt.

Beispiel 3: Dateiänderungsprogramm mit UPDATE, WRITE 8015

STANDARDTEXT KDNR KURZNAME TT.MM.JJ

```
 1     - OPTIONS TIT STANDARDTEXT PHA TEST.

 2     - FILE KUNDEN

 3     - -D
 4     -   SATZ  0 * 140
 5     -   D1          2
 6     -   KDNR        7
 7     -   D2         11
 8     -   NAME       24
 9     -   D3         28
10     -   ORT        20
11     -   D4         43
12     -   UMSATZ      9 2
13     - -I
14     -   FILE  KUNDEN
       -      1 140 SATZ
15     - -C
16     -   DO LOOP
17     -      MAPD NUMMER
18     -      KDNR CHAIN KUNDEN UPD
19     -      MAPD DATEN
20     -      IF CPGFRC = '  '
       -         KDNR UPDAT KUNDEN SATZ
21     -      ELSE
       -         KDNR WRITE KUNDEN SATZ
22     -      END
23     -   END


   Mapname    :  NUMMER
   ======================

   FELD   A F E EC C B ATTFLD ALT.E

   KDNR   A        C


   Mapname    :  DATEN
   ======================

   FELD   A F E EC C B ATTFLD ALT.E


   KDNR   P
   NAME   A        C
   ORT    A
24 UMSATZ A   J
```

Erläuterungen:

Beispiel 3 zeigt ein Änderungsprogramm für eine Kundenstammdatei mit den CPG-Operationen UPDAT und WRITE, die den Programmier- und Speicheraufwand je nach Anwendung reduzieren können (relativ zur Dateiänderung mit EXCPT, vgl. folgendes Beispiel).

1 CPG-Steuerkarte

2 Die Daten der Datei KUNDEN werden dem Data Dictionary entnommen.

3 Die folgenden Felder (in Statement 5 bis 12) werden zu einem Feld mit dem Namen SATZ zusammengefasst. Das Feld ist 140 Stellen groß und alphanumerisch. Für das Feld wird kein Speicherplatz in der TWA reserviert (Eintragung 0 *), sondern es wird von links nach rechts durch die unmittelbar im Anschluss daran definierten Felder belegt. Der Programmierer muss dafür Sorge tragen, dass die Summe der Einzelfeldlängen mit der Feldlänge des überlagerten Feldes übereinstimmt. Bei numerischen Feldern ist dabei die Länge des gepackten Feldes in Bytes und nicht die Stellenzahl für die Berechnung einzusetzen.

14 Von der Datei KUNDEN soll von Stelle 1 bis 140 ein Satz eingelesen werden. Über die Definition der Überlagerung in 3,4 werden dabei gleichzeitig die Felder KDNR, NAME, ORT und UMSATZ gefüllt.

16 Rechenbestimmungen. Der Anfangspunkt einer Endlosschleife wird gesetzt. Eine solche Endlosschleife bewirkt, dass die in ihr enthaltene Statementfolge immer wieder erneut durchlaufen wird; sie kann nur mit der 'Lösch'-Taste verlassen werden.

17 Der erste Befehl innerhalb der Schleife gibt eine QSF-Map mit dem Namen NUMMER aus, um Daten aus ihr einzulesen.

18 Ein Plattensatz wird von der Datei KUNDEN mit dem Schlüssel KDNR gelesen. Der Plattensatz soll bis zum Update gegen weitere Zugriffe von außen gesperrt werden (Service-Funktion UPD). Wenn der Satz nicht vorhanden ist, wird der File Return-Code CPGFRC mit NF für 'not found' gefüllt.

19 Die QSF-Map DATEN wird ausgegeben und wartet auf eine Eingabe.

20 Wenn der Satz gefunden wurde, soll das Feld SATZ in die Datei KUNDEN in den Satz mit dem Schlüssel KDNR zurückgeschrieben werden. Ausgabebestimmungen sind dabei nicht erforderlich, jedoch muss das Feld SATZ in den Eingabebestimmungen unter der Datei KUNDEN beschrieben sein.

21 Wie 20, jedoch wird hier, wenn der Satz beim CHAIN nicht gefunden wurde, ein Satz zur Datei KUNDEN hinzugefügt (CPGFRC ist dann NF)

24 Das Feld Umsatz wird auf dem Bildschirm mit Edit-Code J aufbereitet. Das bedeutet: Führende Nullen werden unterdrückt, die Tausender- und Millionenstellen werden durch einen Punkt abgetrennt, die Dezimalstellen durch ein Komma; ist der Betrag negativ, wird hinter dem Feld ein '-' (Minuszeichen) angezeigt.

Beispiel 4 : Dateiänderungsprogramm mit EXCPT 8017

In modernen Programmen kann der Anwendungsentwickler auf den EXCPT-Befehl ganz verzichten, wenn mit HL1-Datasets gearbeitet wird. Für Spezialisten bleibt EXCPT unverzichtbar, allerdings sollte darauf verzichtet werden, mit Schaltern zu arbeiten oder die Logik in die Output Division auszulagern.

Unter diesem Aspekt sind die beiden ersten unten gezeigten Programme als veraltet anzusehen.

```
STANDARDTEXT                              KDNR KURZNAME        TT.MM.JJ
```

```
 1     - OPTIONS TITEL STANDARDTEXT ROOT PHASE TEST.

 2     - FILE KUNDEN

 3     - -I.  FILE KUNDEN DD

 4     - -C.  DO LOOP
 5     -         MAPD NUMMER
 6     -         KNR CHAIN KUNDEN UPD 20
 7     -         MAPD DATEN
 8     -         EXCPT
 9     -      ENDDO

10     - -O.  FILE KUNDEN DD     ON NOT 20 AND NOT P1
11     -      FILE KUNDEN DD ADD ON 20 AND NOT P1
12     -      FILE KUNDEN    DEL ON NOT 20 AND  P1


   Mapname    :  NUMMER
   ======================

   FELD   A F E EC C B ATTFLD ALT.E

   KDNR   A        C


   Mapname    :  DATEN
   ======================

   FELD   A F E EC C B ATTFLD ALT.E

   KDNR   P
   NAME   A        C
   ORT    A
   UMSATZ A   J
```

Erläuterungen:

6 Im Fall 'Not Found' setzt CHAIN hier den Schalter 20.

8 Durch den Befehl EXCPT wird in die Output-Division verzweigt. Damit ist die Beschreibung der Ausgabe in die Output-Division verla-

gert.

Es werden grundsätzlich alle Bestimmungen der Output Division ausgeführt, es sei denn, sie sind über Bedingungsschalter oder Namen verriegelt.

Durch Angabe von bis zu drei Schaltern oder einem Namen hinter dem EXCPT könnte auch schon in der Procedure Division eine Vorauswahl der auszuführenden Ausgabebestimmungen getroffen werden.

Folgende Programmänderung führt zum gleichen Ergebnis (Auszug):

```
      :
 7    -         MAPD DATEN
 8A   -         ON P1     NOT 20  EXCPT  LÖSCH
 8B   -         ON NOT P1 NOT 20  EXCPT  ÄNDER
 8C   -         ON NOT P1     20  EXCPT  HINZU

10    - -O.  FILE KUNDEN DD      ÄNDER
11    -      FILE KUNDEN DD ADD  HINZU
12    -      FILE KUNDEN    DEL  LÖSCH
```

10 Auf der Datei KUNDEN soll ein Satz geändert werden. Die Namen der zu ändernden Felder werden dem Data Dictionary (DD) entnommen.

11 Wie 10, jedoch soll zur Datei ein neuer Satz hinzugefügt werden. Dazu ist das Schlüsselwort ADD erforderlich, die Reihenfolge muss bei den Schlüsselworten eingehalten werden.

12 Wie 10 und 11. Zum Löschen eines Datensatzes ist das Schlüsselwort DEL erforderlich.

Folgende weitere Änderung führt ebenfalls zum gleichen Ergebnis:

```
      :
 6    -         KNR CHAIN KUNDEN UPD
 7    -         MAPD DATEN
 8A   -         IF CPGFRC = 'NF'
 8B   -            ON NOT PF1  EXCPT  HINZU
 8C   -         ELSE
 8D   -            IF CONDITION PF1
 8E   -               EXCPT  LÖSCH
 8F   -            ELSE
 8G   -               EXCPT  ÄNDER
 8H   -            ENDIF
 8I   -         ENDIF
      - -O.  FILE KUNDEN DD      ÄNDER
      -      FILE KUNDEN DD ADD  HINZU
      -      FILE KUNDEN    DEL  LÖSCH
```

6 Auf den Schalter (20) kann man verzichten. Die Information 'gefunden' bzw. 'nicht gefunden liefert das interne Feld CPGFRC.

8A 'NF' steht für 'Satz beim CHAIN nicht gefunden (not found).

Beispiel 5: RRDS-Datei mit numerischem Keyfeld 8020

RRDS-Dateien werden analog zu KSDS-Dateien verarbeitet.

Lesen ist sowohl sequentiell als auch im direkten Zugriff möglich. Veränderungen sind in der Form Update, Hinzufügen und Löschen eines Satzes unterstützt, d.h. es kann beim CHAIN auch festgestellt werden, ob ein Satz vorhanden ist oder nicht.

Die einzelnen Funktionen sind im Folgenden anhand von Programmfragmenten erläutert:

```
1   - OPTIONS  TITEL RRDS#BEISPIEL  PHASE TEST.

2   - FILE RRDS
3   - -D.  RRN 9 0.                    * Relative Satznummer
4   - -I.  FILE RRDS. 1 100 SATZ
```

Sequentielles Lesen:

```
5   - -C.  DO UNTIL CPGFRC = 'EF'
6   -         1 READ RRDS
7   -      ENDDO
8   -      RANDOM RRDS
```

Sätze hinzufügen

```
 5  - -C. DO LOOP
 6  -        RRN = RRN + 1
 7  -        RRN CHAIN RRDS
 8  -        IF CPGFRC = 'NF'.        * Not found
 9  -           EXCPT NEU
10  -        ENDIF
11  -     ENDDO
```

Direkter Zugriff:

```
5   - -C. RRN = 200
6   -     RRN CHAIN RRDS
```

Update:

```
5   - -C. RRN  CHAIN  RRDS  UPD
6   -     IF CPGFRC = '  '
7   -        EXCPT UPDATE
```

Löschen eines Datensatzes:

```
5   - -C. RRN  CHAIN  RRDS
6   -     DELET RRDS
```

Zugehörige Output-Division:

```
    - -O. FILE RRDS ADD NEU.  SATZ 100
    -     FILE RRDS   UPDATE. SATZ 100
```

Beispiel 6: ESDS-Datei 8025

```
- OPTIONS TIT ESDS#-#BEISPIELPROGRAMM PHASE TEST.

- FILE ESDS

- -D
-     ARBA 4
-     KEY  4
-     RBAN 9 0

- -I
-     FILE ESDS
-        1  100   SATZ
```

Hinzufügen eines Satzes. Nach dem Hinzufügen wird die aktuelle RBA (relative Byte-Adresse) im Feld CPGKxx (xx = lfd. Nr. der Datei in der Files Division) zur Verfügung gestellt:

```
- -C. EXCPT NEU.               * Satz hinzufügen
-     MOVE  CPGK01 TO ARBA.    * Aktuelle RBA retten
```

Direkter Zugriff. Ist das Schlüsselfeld alphanumerisch, so muss es mit einem numerischen Feld binär gefüllt werden (EDIT). Bei numerischem Schlüssel wird die Aufbereitung intern vorgenommen. Mit dem numerischen Feld RBAN könnte im Beispiel direkt zugegriffen werden.

```
- -C. EDIT KEY
-     KEY CHAIN ESDS
```

Update:

```
- -C. EDIT KEY.
-     KEY CHAIN ESDS
-     IF CPGFRC = '  '.        * Satz gefunden
-        EXCPT ALT
-     END
```

Sequentiell lesen:

```
- -C. FILL X'00' KEY.          * Erste RBA, alternativ :  RBAN = 0
-     DO UNTIL CPGFRC = 'EF'
-        KEY READ ESDS
-     END
-     RANDOM ESDS
```

Zugehörige Output-Division:

```
- -O. FILE ESDS  ADD  NEU
-        SATZ  100
-     FILE ESDS       ALT
-        SATZ  100
-     FIELD KEY
```

```
    -        RBAN    4   BINÄR
10a - **     LRBA  4.              * Vorheriger Satz
10b - **    4 '00000064' HEX.      * 2. Satz (bei Satzlänge 100 Byte)
10c - **    4 '00001000' HEX.      * 41. Satz bei 4K CI-Size
```

Beispiel 7 : Hinzufügen in einer ESDS-Datei 8030

Es wird die Möglichkeit geboten, die zehn zuletzt hinzugefügten Sätze zu durchblättern

```
 1  - OPTIONS ROOT PHASE TEST.
 2  - FILE CPGESD
    - -D
 3  -    FG  10 * 4
 4  -    I  3 0
 5  -    KEY  4
    - -I
 6  -    FILE CPGESD
 7  -       6 10 SATZ
    - -C
 8  -   DO LOOP
 9  -     MAPD MAP1
10  -     IF CONDITION DE
11  -        EXCPT.                          * Ausgabe auf Datei
12  -        CPGK01 CHAIN CPGESD  CHECK 99. * siehe unten !
13  -        ROLL-BACK FG.
14  -        MOVE  CPGK01 TO FG(1)
15  -        I = 0
16  -     ELSE
17  -        IF CPGMPF = 'P1'.           * Taste PF1
17  -           I = I + 1.               * zum Rückwärtsblättern
17  -        ELSE
18  -           IF CPGMPF = 'P2'.        * Taste PF2
18  -              I = I - 1.            * zum Vorwärtsblättern
18  -           END
    -        END
19  -        IF I > 0
20  -           IF I <= 10
21  -              KEY = FG(I)
22  -              IF KEY >< '    '
23  -                 KEY CHAIN CPGESD 99
24  -              ENDIF
25  -           ENDIF
26  -        ENDIF
27  -     ENDIF
28  -   ENDDO
29  - -O
29  -   FILE CPGESD ADD
30         SATZ  10
```

Erklärung des Programmteils 'NEUER-SATZ', Statements 10-15 und 29,30:

Der Satz, der aus der Map eingelesen wurde, wird der ESDS-Datei hinten hinzugefügt.

Das CHAIN in Statement 12 wird durchgeführt, damit der Satz tatsächlich auf die Platte geschrieben wird; ohne das CHAIN wäre der neue Satz zu diesem Zeitpunkt nur im Speicher hinzugefügt. Mit der Servicefunktion CHECK im CHAIN-Befehl erreicht man, dass keine Daten eingelesen werden.

Die Feldgruppe FG nimmt die RBAs der letzten zehn hinzugefügten Sätze auf. Mit dem ROLL-BACK werden die bereits geretteten RBAs in der Feldgruppe nach hinten geschoben (Statement 12). Die aktuelle RBA wird im ersten Element der RBA-Feldgruppe abgestellt.

Die Bezugszahl 99 in den beiden CHAIN-Statements ist erforderlich,weil der Compiler entweder die Angabe einer Bezugszahl oder die Abfrage des Return-Codes im internen Feld CPGFRC verlangt.

Beispiel 8 a : Drucken im Linemode (veraltet , wird heute mit) (CPG4 programmextern gelöst) 8035

```
- OPTIONS PHASE TEST.

- FILES BILD L86C.

- FORMS L86C LENGTH 72  ZEILE 10 KANAL 1  Z 35 K 2  Z 70 K 12

- -C. EXCPT

- -O. FILE BILD. 550 'BEISPIEL DRUCKER'. 650 'L I N E M O D E '
-     FILE L86C SPACE 3 5  SKIP 01.
-                                   30 'VORSCHUB VOR DEM DRUCKEN'
-                                   60 'NACH KANAL 01 = ZEILE 10'
-                                   90 '3 ZEILEN VOR DEM DRUCKEN'
-                                  120 '5 ZEILEN NACH DEM DRUCK.'
-     FILE L86C SPACE 2 1  SKIP 02.
-                                   30 'VORSCHUB VOR DEM DRUCKEN'
-                                   60 'NACH KANAL 02 = ZEILE 35'
-                                   90 '2 ZEILEN VOR DEM DRUCKEN'
-                                  120 '1 ZEILE  NACH DEM DRUCK.'
-  FILE L86C SPACE # 2.
-                                  120 '2 ZEILEN NACH DEM DRUCK '
-  FILE L86C SPACE # 1 SKIP 12.
-                                   30 'VORSCHUB VOR DEM DRUCKEN'
-                                   60 'NACH KANAL 12 = ZEILE 70'
-                                  120 '1 ZEILE  NACH DEM DRUCK.'
```

Der Line-Mode-Drucker im Programm heisst L86C. In der Forms-Bestimmung ist die Formularlänge mit 72 Zeilen angegeben. Weiterhin sind Kanäle vereinbart, und zwar die Zeilen 10, 35 und 70 als Kanäle 1, 2 und 12.

Der Rest des Programms erklärt sich durch die Textkonstanten in der Output-Division selbst.

Es ist empfehlenswert, ähnlich der Auslagerung der Bildschirmein-/ -ausgabe mit QSF die Druckausgabe ebenfalls programmextern zu programmieren. Dazu steht das Lattwein-Produkt QTF (Quick Text Facility) zur Verfügung.

Beim Einsatz von QTF entfallen die Eintragungen in der Output Division für die Druckausgabe vollständig.

Beispiel 8 b : Drucken im Buffermode

1. Bei Einsatz von QSF:

Bilder, die für die Bildschirmein-/-ausgabe schon im QSF beschrieben sind, können ohne zusätzlichen Aufwand auch auf den Drucker ausgegeben werden. Dazu steht der Befehl MAPP zur Verfügung, der eine Map auf einen frei wählbaren Online-Drucker ausgibt.

2. Ohne Einsatz von QSF:

Die Druckausgabe wird wie eine Bildschirmausgabe programmiert. Die Dateibestimmung muss in diesem Fall eine entsprechende Eintragung haben. Die Ausgabe erfolgt über die Operation EXCPT, die in die Output Division verzweigt; dort ist die Ausgabe wie eine Bildschirmausgabe zu programmieren.

Es ist bei der Programmierung im Buffermode zu beachten, dass der Ausdruck optimiert wird. Für den 'Ausdruck' einer Leerzeile muss mindestens ein Blank in dieser Zeile ausgegeben werden.

Diese Regel ist immer zu beachten, unabhängig davon, ob QSF eingesetzt wird oder nicht.

Beispiel 9: Feldaufbereitungen mit EDIT 8040

```
OPTIONS ROOT PHASE TEST.

FILE KUNDEN

DATA DIVISION.
   BZ  20 * 78.               * Bildschirmzeile
   I  3 0.                    * Index
   LAND  3.                   * Land
   ORT  23.                   * Ort
   PLZ  5.                    * Postleitzahl
   STADT  35.                 * Anschrift (letzte Zeile)

INPUT DIVISION
   FILE KUNDEN DD

PROCEDURE DIVISION.
   :
   DO UNTIL EC ><  '  '
     KEY READ KUNDEN
     IF CPGFRC = 'EF'
        EC = 'EF'.                  * Abbruchkriterium der Schleife
     ELSE
        IF LAND = '   '
           EDIT STADT TYPE INLAND.  *  <==  EDIT mit Typ
        ELSE
           EDIT STADT TYPE AUSLAND. *  <==  EDIT mit Typ
        ENDIF
        IF WERT < MIN
           I = I + 1
           EDIT BZ(I).              *  <==  indiziertes EDIT
           IF I >= 20
              EC = '20'.            * Abbruchkriterium der Schleife
           ENDIF
        ENDIF
     ENDIF
   ENDDO
   MAPO  MASKE3
   :

OUTPUT DIVISION
   FIELD BZ.                        * Feldgruppenelemente editieren
      FIRMA 30
      STADT 66
      WERT  78 EDITCODE K
   FIELD STADT  TYPE  INLAND
      PLZ    5
      ORT   29
   FIELD STADT  TYPE  AUSLAND
      LAND   3
             5 '-'.                 * Trennzeichen Land/Postleitzahl
      PLZ   11
      ORT   35
```

Das Beispiel zeigt die Aufbereitung von Feldern über die Output-Division. In der Output-Division wird mit dem Schlüsselwort FIELD und

dem Feldnamen die Aufbereitungsvorschrift hinterlegt.

Auch bei indiziert verarbeiteten Feldgruppen (wie hier bei BZ) wird nur der Feld(gruppen)name angegeben.

Soll ein Feld auf verschiedene Arten aufbereitet werden können, so arbeitet man in der Procedure Division und entsprechend in der Output-Division mit EDIT-Typen (wie am Beispiel STADT gezeigt).

Das Feld STADT enthält die letzte Zeile der Anschrift, unterschiedlich aufbereitet für Inlands- und Auslandsadressen, z.B.

```
52477 Düren                      bei Inland, aber
  A - 1050  Wien                 bei Ausland
```

Beispiel 10: READ-BACK 8045

READ-BACK KDNR KURZNAME TT.MM.JJ

```
 1  - OPTIONS ROOT PHASE TEST
    -         TITEL READ-BACK
    -         END
 2  - FILE KUNDEN
    - -D
 3  -    PAGE  20 * 70
    - -I
 4  -    FILE KUNDEN DD
    - -C
 5  -    FILL '9' TO KDNR
 6  -    DO LOOP
 7  -      KDNR READB-PAGE KUNDEN PAGE
 8  -      MAPD BILD
 9  -    END
10  - -O
11  -    FIELD PAGE
12  -       KDNR  7
13  -       NAME  33
14  -       ORT  55
```

```
   Mapname    :  BILD
   ======================

   FELD   A F E EC C B ATTFLD ALT.E

   PAGE   P
```

Erläuterungen:

3 Eine Feldgruppe mit dem Namen PAGE wird definiert. Die Feldgruppe besteht aus 20 Feldern mit einer Länge von je 70 Bytes. Sie soll eine Bildschirmseite mit 20 Zeilen aufnehmen.

5 Das Feld KDNR wird mit dem Wert '9999999' gefüllt. Es wird dabei

angenommen, dass auf der Datei KUNDEN ein Satz mit dem Schlüssel '9999999' existiert. Voraussetzung für den Befehl READ-BACK ist, dass das Schlüsselfeld beim Zugriff einen existierenden Schlüsselwert enthält.

6 Von der Datei 'KUNDEN' werden so viele Sätze gelesen, wie die im Ergebnisfeld eingtragene Feldgruppe Elemente enthält. Die Seite wird in der Feldguppe PAGE gespeichert. Die Datei KUNDEN wird durch den Befehl READ-BACK rückwärts verarbeitet. Der erste gelesene Satz ist der, dessen Schlüssel gleich dem Inhalt des Feldes KDNR ist.

Anschließend wird die PAGE in der QSF-Map BILD ausgegeben; nach der Anzeige wird die PAGE erneut mit Daten aus der Datei KUNDEN gefüllt, diese wiederum angezeigt usw.

11 Jede Zeile der Bildschirmseite wird aufbereitet wie in Zeile 7 beschrieben. Ausgelöst wird die Aufbereitung hier durch die Operation READ-BACK.

12 Das Feld KDNR wird in Position 1 bis 7 aller Elemente der Feld-
13 guppe ausgegeben. Analog werden die Felder NAME und ORT mit ihrem
14 letzten Byte auf die Positionen 33 und 55 der Feldgruppenelemente ausgegeben.

Beispiel 11: Update VSAM variable Satzlänge 8050

1. Möglichkeit: Die Länge des Ausgabesatzes wird bestimmt durch die höchste Ausgabeposition in der Output Division.

```
 1  - OPTIONS  PHASE TEST.
 2  - FILE DATEI
 3  - -I. FILE DATEI. 1  10 KEY

 4  - -C. :
10  -     KEY CHAIN DATEI
11  -     ON PF1 EXCPT LG50
12  -     ON PF2 EXCPT LG100

50  - -O. FILE DATEI ALG  LG50.   50 'SATZLÄNGE 50'
51  -     FILE DATEI ALG  LG100. 100 'SATZLÄNGE 100'
```

Erläuterungen:

10 Mit dem Befehl CHAIN wird auf die Datei DATEI zugegriffen.

11, 12 Gesteuert durch Programmfunktionstasten, die vom Bildschirm eingelesen werden, sollen auf die DATEI mit variabler Satzlänge Sätze unterschiedlicher Länge ausgegeben werden : Bei PF1 soll der ausgegebene Satz 50, bei PF2 100 Bytes lang sein.

50, 51 Diese Verarbeitungsart ist nur möglich, wenn in den Dateibestimmungen der Output-Division das Schlüsselwort ALG angegeben wird.

Außerdem muss natürlich die Dateibeschreibung in der Files-Division oder programmextern die Eintragung für variable Satzlänge enthalten.

2. Möglichkeit: Die Länge des Ausgabesatzes wird bestimmt durch den Inhalt des internen Feldes CPGVRL.

```
 4  - -C. :
10  -     KEY CHAIN DATEI
11  -     ON PF1  CPGVRL = 50
12  -     ON PF2  CPGVRL = 100

50  - -O. FILE DATEI ALG VARIABEL
51  -        CPGVRL  50  EDIT  Z
52  -                46 'Satzlänge'
```

Erläuterung:

50 Für diese Art der Verarbeitung muss in der Satzbestimmung der Output Division zusätzlich zu ALG das Keywort VAR codiert werden.

Beispiel 12: Cursor-Stop (Beispiel ist veraltet !) 8051

```
1   - OPTIONS TIT CURSOR-STOP PHA TEST.
2   - FILE BILD
3   - -I.  FILE BILD.  114 118 EIN1.  214 218 EIN2.  314 318 EIN3

4   - -C.  DO LOOP. EREAD BILD. END

5   - -O.  FILE BILD. 110 '1. EINGABE'.  EIN1 118  ATTR A  CURSOR
6   -                 210 '2. EINGABE'.  EIN2 218  ATTR A
7   -                 310 '3. EINGABE'.  EIN3 318  ATTR A
8   -                 319  ATTRIBUT 0
```

Erläuterungen:

8 Eine Stelle hinter der letzten Eingabe wird das Bildschirmattribut '0' (Null) ausgegeben.

Der Cursor springt nach der dritten Eingabe eine Stelle nach rechts und die Tastatur wird für weitere Eingaben blockiert.

Mit der Sprungtaste kann der Cursor zu einem Eingabefeld bewegt werden, um eine fehlerhafte Eingabe zu korrigieren.

Durch den Cursor-Stop wird verhindert, dass durch irrtümliches Betätigen einer Taste die ersten Bildschirmfelder überschrieben werden.

Beispiel 13: Temporary Storage Queuing 8060

```
 1  - OPTIONS PHA TEST.
 2  - FILE STOR UPD QUEUE FIX 80 INDEPENDENT STORAGE
 3  - -D
 4  -      PAGE 16 * 78
 5  -      I  3 0
 6  - -I
 7  -      FILE STOR
 8  -         1 78 ZEILE
 9  - -C
10  -      DO 16 TIMES WITH I
11  -        I  READ  STOR
12  -        IF CPGFRC = 'EF'.
13  -          BREAK
14  -        ELSE
15  -          PAGE(I) = ZEILE
16  -        END
17  -      END
18  -      MAPD  BILD
19  -      PURGE STOR
20  -      DO 16 TIMES WITH I
21  -        ZEILE = PAGE(I)
22  -        EXCPT
23  -      ENDDO
24  -      EXITI 'TEST'
25  - -O
26  -      FILE STOR ADD
27  -        ZEILE  78
```

```
Mapname    :  BILD
======================

FELD   A F E EC C B ATTFLD ALT.E

PAGE   P
```

Erläuterungen:

Das Beispiel zeigt die Verarbeitung einer Temporary Storage Queue.

2 Dateizuordnung für Temporary Storage Queuing. Das Schlüsselwort UPD bedeutet, dass der Storage Bereich sowohl für Ein- als auch für Ausgabe benutzt wird. Die Eintragung QUEUE ist für Temporary Storage Queuing erforderlich. Die Satzlänge ist fest (FIX) und 80 Stellen groß. INDependent bedeutet, dass der Bereich bildschirmunabhängig ist, also für alle Bildschirme verfügbar. Der Eintrag STORAGE definiert die Datei als TS-Bereich.

Alle Einträge sollten aber nicht im Programm eingegeben werden, sondern im Data Dictionary File.

10 Mit der DO-Operation wird eine Schleife begonnen,mit der ein Index I jeweils um 1 erhöht wird. Die Schleife wird 16 mal durchlaufen.

Mit dem Index I wird die Temporary Storage Queue STOR gelesen; der Zugriff erfolgt hierbei (anders als bei sonstigen Dateizugriffen)

direkt auf den Satz mit der laufenden Nummer I.

Um beim nächsten READ den folgenden Satz zu lesen, muss der Index jeweils um 1 erhöht werden (wozu hier die DO- Schleife verwendet wird). Es wäre jedoch auch möglich, die folgenden READs ohne Angabe eines Schlüsselfeldes durchzuführen - hierdurch würde auch ohne Erhöhung des Indexfeldes jeweils der nächste Satz gelesen.

12 In CPGFRC wird 'EF' abgestellt, wenn das Ende des Storagebereichs erreicht. 'EF' wird auch gesetzt, wenn bei der READ-Operation kein Storage gefunden wird.

15 Die DO-Schleife wird nur dann bis zu ihrem Ende durchlaufen, wenn End of File nicht gesetzt wird. Ansonsten beendet das BREAK die DO-Schleife.

19 Mit der Operation PURGE wird der Storage-Bereich STOR gelöscht.

22 Ausgabe eines Satzes in den Storage-Bereich (siehe auch 23).

23 Mit der Operation EXITI wird das gleiche Programm (TEST) wieder gestartet. Dieser Weg kann anstelle eines DO-Loops, der das gesamte Programm umfasst, beschritten werden, um zu erreichen, dass vor der erneuten Ausführung alle Felder des Programms auf Null bzw. Blank initialisiert werden.

26 Der Eintrag ADD bedeutet, dass Sätze hinten an den Bereich angehängt werden.

Beispiel 14: Variable Cursorposition, z.B. bei Fehlermeldungen 8065

```
:

- PROCEDURE DIVISION.
-  IF ERROR = '**'
-     CPGMCU = 'KDNR  '

-  END
-  MAPD MASKE
:
```

Erläuterungen:

Das Beispiel zeigt das Prinzip der variablen Positionierung des Cursors in einer mit QSF erstellten Map.

Zur Cursor-Positionierung steht das CPG-interne Feld CPGMCU zur Verfügung. Es ist 6 Bytes groß und alphanumerisch.

Mit der Zuweisungsoperation wird in das Feld CPGMCU der Name des Feldes übertragen, auf dem der Cursor bei der nächsten Ausgabe stehen soll. Der Feldname wird dabei als Textkonstante in Hochkommata übertragen.

Ist das Feld CPGMCU nicht gefüllt, so wird der Cursor dort positioniert, wo er in der Beschreibung der Map angegeben wurde.

Zu beachten ist, dass das Feld CPGMCU nach jeder Bildschirmausgabe gelöscht wird; sind also in einem Programm mehrere Bildschirmausgaben vorhanden, so muss der Programmierer sicherstellen, dass zur Zeit einer MAP-Ausgabe-Operation das Feld CPGMCU 'richtig' gefüllt ist.

Handelt es sich bei dem Feld, in das der Cursor positioniert werden soll, um eine Feldgruppe, dann kann der Index des gewünschten Elements dem Feld CPGMCI (dreistellig numerisch) zugewiesen werden

Beispiele zur Operation FIND: 8070

Die Anwendung der Operation FIND setzt voraus, dass eine Tabelle angelegt wurde, die bei der ersten Ausführung der Operation FIND zur Verarbeitung in den Hauptspeicher geladen wird.

In den folgenden Beispielen zur FIND-Operation wird vorausgesetzt,dass eine solche Tabelle generiert ist; sie habe folgenden Aufbau:

```
Spalte  1 -  6  Auftragsnummer
Spalte  7 - 12  Auftragsdatum
Spalte 13 - 19  Kundennummer
Spalte 20 - 29  Artikelnummer
Spalte 30 - 31  Vertreternummer
Spalte 32 - 34  Warengruppe
```

Die Tabellengröße ist also 34, die 'Schlüssellänge' 10, weil jedes Element Schlüssel sein kann und die Artikelnummer mit 10 Bytes der längste mögliche Schlüssel ist.

Beispiel 15: RNDOM bei der Tabellenverarbeitung

```
        :
20      - AUFTNR FIND VIEW
        - IF CONDITION NOT EOF
21      -    EXSR UP01
        - ENDIF
22      - RANDOM VIEW
23      - ARTNR  FIND VIEW
        :
```

20 Die Tabelle wird nach einer Auftragsnummer durchsucht. Wird diese nicht gefunden, so wird der Schalter EF gesetzt. Das folgende FIND beginnt wieder beim ersten Element der View mit der Suche. Wird die Auftragsnummer aber in der Tabelle gefunden, so wird beim folgenden FIND an dieser Stelle der Tabelle wieder aufgesetzt.

22 In diesem Fall ist ein RANDOM VIEW erforderlich, um den internen Zeiger auf das erste Element der View zu setzen.

Beispiel 16: Anzeige einer Tabelle nach Auswahlkriterium 8071

Es sollen genau die Aufträge angezeigt werden, die einem bestimmten Vertreter zugeordnet sind.

```
 1      - FILE VIEW  INP  FIX  34 10 TABLE
 2      - -D
 3      -      PAGE  20 * 78
 4      -      I  3 0
 5      - -I.
 6      -      FILE  VIEW
 7      -         1  6   AUFTNR
 8      -         7 12 0 DATUM
 9      -        13 19   KDNR
10      -        20 29   ARTNR
11      -        30 31   VNR
12      -        32 34   WGRP
13      - -C.  DO LOOP
14      -         MAPD EINLESEN
15      -         DO 20 TIMES WITH I
16      -            VNR FIND VIEW
17      -            IF CPGFRC = 'EF'
18      -               BREAK
19      -            ELSE
20      -               EDIT PAGE(I)
21      -            ENDIF
22      -         ENDDO
23      -         MAPD ANZEIGE
24      -      END
25      - -O
26      -     FIELD PAGE
27      -        AUFTNR  6
28      -        DATUM   16  EDITCODE Y
29      -        KDNR    30
30      -        ARTNR   76
```

Erläuterungen:

1 und 6 - 12. Beschreibung der Datenview. Die Beschreibung der View und ihrer Struktur sollte allerdings nicht im Programm, sondern im Data Dictionary vorgenommen werden.

14 Einlesen einer Vertreternummer (VNR)

16 Die Tabelle wird nach der eingegebenen Vertreternummer durchsucht. Wird die Nummer gefunden, so werden die zugehörigen Tabellenelemente eingelesen. Beim nächsten Durchlaufen der Schleife wird beginnend mit dem folgenden Tabellenelement der Rest der Tabelle durchsucht. Mit FIND kann die Tabelle vollständig sequentiell durchlaufen werden.

20 Die Elemente der Page werden nur mit Feldern aufbereitet, die in der Tabelle gefunden wurden.

23 Die gefüllte Bildschirmseite PAGE wird in der Map ANZEIGE ausgegeben.

Beispiel 17: Aufträge anzeigen 8072

Aufträge anzeigen, dabei zusätzliche Informationen wie Klartexte von Kunden und Artikeln aus Dateien anziehen.

```
 1      - FILE VIEW
 2      - FILE KUNDEN
 3      - FILE ARTSTA
        - -D.
 4      -     FEHLER  26
 5      -     I  3 0
 6      -     PAGE  20 * 78
        - -I
 7      -    FILE  VIEW  DD
 8      -    FILE KUNDEN
        -       37  61  KUNDE
 9      -    FILE ARTSTA
        -       21  45  ARTIKL
        - -C
10           :
11      -    AUFTNR FIND  VIEW
12      -    IF CPGFRC = 'EF'
13      -       EXSR FEHLER
14      -    ELSE
15      -       KDNR CHAIN KUNDEN
16      -       IF CPGFRC = '  '
17      -          ARTNR CHAIN ARTSTA
18      -          IF CPGFRC = 'NF'
19      -             FEHLER = 'Artikel nicht gefunden'
20      -          END
21      -       ELSE
22      -          FEHLER = 'Kunde nicht gefunden'
23      -       END
24      -    END
        :
25      -    EDIT PAGE(I)
        :

26      - FIELD PAGE
27      -    AUFTNR   6
28      -    DATUM   16 EDIT Y
29      -    ARTIKL  44
30      -    KUNDE   72
```

Erläuterungen:

1 7 Beschreibung der Datenview und ihrer Struktur wird dem Data Dictionary entnommen (Werte wie im vorhergehenden Beispiel)

11 Die Tabelle ist nach Auftragsnummern geordnet. Mit dem Befehl FIND kann in der Tabelle direkt auf einen Auftrag zugegriffen werden. Liegt der FIND-Befehl in einer Schleife, so werden (von einer bestimmten Auftragsnummer an) alle Tabellenelemente eingelesen.

15 Mit der in der Tabelle gefundenen Kundennummer wird auf die Kundendatei zugegriffen, um den Namen des Kunden als Klartext einzulesen. Voraussetzung ist allerdings, dass das Tabellenelement KDNR dem Schlüsselfeld der Datei KUNDEN entspricht; dies ist bei

Generierung der Datenview zu berücksichtigen.

17 Mit der in der Tabelle gefundenen Artikelnummer wird auf die Artikelstammdatei zugegriffen, um die Artikelbezeichnung als Klartext einzulesen. Voraussetzung ist allerdings, dass das Tabellenelement ARTNR dem Schlüsselfeld der Datei ARTSTA entspricht; dies ist bei Generierung der Datenview zu berücksichtigen.

26 Die Page wird sowohl mit Werten aus der Tabelle als auch mit zusätzlichen Informationen, die durch Dateizugriffe bereitgestellt wurden, aufbereitet.

Beispiel 18: Variabler Mapname 8075

Zur Flexibilisierung der MAP-Befehle steht der variable Mapname zur Verfügung. In diesem Zusammenhang kann nicht nur der Name der Map variabel gehandhabt werden, sondern auch das Löschen des Bildschirms vor der Ausgabe, der Write Control Character und die Minimierung der Datenübertragung bei entfernt betriebenen Bildschirmen.

Die Syntax und einige Verarbeitungsbeispiele mit variablem Mapnamen sind im folgenden beschrieben:

Die Schnittstelle zum QSF bildet ein 16-stelliges Alphafeld; in den ersten acht Stellen enthält es den Mapnamen, in der neunten die Information über das Löschen des Bildschirms; die zehnte Stelle enthält den WCC und in der elften kann die Übertragung der Konstanten unterdrückt werden; die restlichen fünf Stellen sind derzeit noch Reservebytes. Ein solches Feld kann zum Beispiel in der Data Division als Überlagerung beschrieben werden:

```
- -D.
-   MAPNAM  8.    * Maskenname
-   ERASE  1.     * Bildschirm vor der Maskenausgabe löschen ?
-   WCC  1.       * Write Control Character
-   OCTL 1.       * Nur variable Felder ausgeben ?
-   FLDCLR 1.     * Mapfelder oder variable Attribute im Pro-
-                 * gramm löschen ? (= kein Map-Input/-Output)
-   ORG MAPNAM.   * Redefinieren der Speichers ab MAPNAM
-     MAPCTL 16.  * 16-stelliges Map Control-Feld für variable
-                 * Mapverarbeitung
-   DRUC  4
```

Beispiel:

Es soll die Map BUCH01 ausgegeben werden. Vor der Ausgabe soll der Bildschirm gelöscht werden, bei der Ausgabe soll die Hupe ertönen. Anschließend soll wahlweise (gesteuert durch die Taste PF11) diese Map auf dem Drucker DR01 ausgegeben werden:

```
- -C. MAP   = 'BUCH01  '
-     ERASE = 'Y'.          * auf jeden Fall Erase
-     WCC   = 'H'.          * Ausgabe mit Hupe
-     DRUC  = 'DR01'        * Druckernamen vorgeben
-     MAPO-VAR MAPCTL
-     ON PF11   DRUC  MAPP-VAR  MAPCTL
```

Der variable Mapname ist natürlich für alle MAP-Operationen unterstützt.

Bei der Verwendung des variablen Mapnamens ist der Programmierer dafür verantwortlich, dass das 16-stellige Feld mit sinnvollen Werten gefüllt ist. Ein Mapname muss immer angegeben werden; die restlichen Informationsbytes sind optional zu füllen; bleiben sie leer, so wird die im QSF angegebene Information übernommen, ansonsten wird sie überschrieben.

Beispiel 19: TWA-SAVE und TWA-LOAD 8080

Die beiden Operationen TWASV und TWALD erleichtern das Zwischenspeichern von Daten. Eine typische Anwendung: Bei transaktionsorientiert geschriebenen Programmen können mit Hilfe dieser Operationen auch solche Daten einfach an eine folgende Transaktion übergeben werden, die nicht auf dem Bildschirm zwischengespeichert werden.

```
      :
      - -C.
      -      TWA-LOAD BSP1
      -      MAP BEISPIEL
      :
      :   *  Verarbeitung der eingelesenen Daten.  Nicht alle Daten,
      :   *  die  für die  weitere  Verarbeitung  benötigt werden,
      :   *  sind in der Map BEISPIEL enthalten.
      :
      -      MAPO BEISPIEL
      -      TWA-SAVE BSP1
      -      EXITT  'BSP1'
```

Es handelt sich im Beispiel um eine Transaktion, die sich immer wieder selbst aufruft. Zu Beginn des Programms wird die gesamte TWA der vorherigen Transaktion geladen. Somit sind dem Programm auch die Daten bekannt, die nicht über den Bildschirm übergeben wurden. Voraussetzung ist dabei, dass die TWA vor dem Verlassen des Programms auch mit TWA-SAVE unter dem gleichen Namen (hier: BSP1) gerettet wurde.

TWA-SAVE und TWA-LOAD können nur in solchen Programmen eingesetzt werden, die sich selbst wieder aufrufen.

Beispiel 20: Lesen von Dateien mit Satzlänge größer 8 K 8090

Dateien mit Satzlängen größer 8 K können über variable Eingabepositionen in Verbindung mit dem CPG-internen Feld CPGFIS eingelesen werden.

Die variable Verarbeitung der Eingabepositionen wird in der Satzbestimmung der Input Division durch den Parameter VAR erreicht, der Verschiebefaktor wird in der Procedure Division im internen Feld CPGFIS (5-stellig numerisch) gepflegt.

```
-  OPTIONS PHASE TEST.
-  FILE BIGFILE INP FIX  9999 10  KSDS
-  FILE BIGFIL2 INP FIX  9999 10  KSDS

- -I.
-     FILE BIGFILE.
-          15   24  F0
-        1001 1010  F1000
-     FILE BIGFILE  VAR
-           1   35  F10000
- -C.
:
-  CPGFIS = 10000
-  ' ' READ BIGFILE
:
```

Angenommen wird hier, dass die Datei BIGFILE 12.000 Byte groß ist.

Die Felder F0 und F1000 werden auf herkömmliche Art gelesen. Zum Lesen der Information von Stelle 10001 bis 100035 wird der Verschiebefaktor benötigt; im Programm wird er durch CPGFIS = 10000 gesetzt. Die Verschiebung der Eingabepositionen ist nur möglich, weil an die Satzbestimmung der Input Division das Schlüsselwort VAR angehängt wurde.

Beispiel 21a: Programm zur Dateipflege, dialogorientiert (veraltet mit Schaltern und GOTO-Sprüngen) 8101

```
 1  - OPTIONS TITEL Dateipflege#im#Dialog PHA TEST.

 2  - FILE CPGWRK

 3  - -D.  PAGE 20 * 78.  I  3 0.

 4  - -I.  FILE CPGWRK DD

 5  - -C.
 6  -  A100
 7  -      MAPD MASKE1
 8  -      ON P1  GOTO A300
 9  -  A200.                          * Datei sequentiell anzeigen
10  -      DO 20 TIMES WITH I
11  -        KEY READ CPGWRK
12  -        ON EF  GOTO A250
13  -        EDIT PAGE(I)
14  -      ENDDO
15  -      KEY READ CPGWRK
16  -  A250.
17  -      RANDOM CPGWRK
18  -      MAPD MASKE2
19  -      IF CON P2. GOTO A100
20  -      ELSE.      GOTO A200
21  -      ENDIF
22  -  A300.                          * Datensätze ändern/hinzufue-
23  -                                 *             gen/löschen
24  -      KEY CHAIN CPGWRK  50
25  -      MAPD MASKE3
26  -      EXCPT
27  -      GOTO A100

28  - -O.  FILE CPGWRK DD       ON DE AND NOT 50
29  -      FILE CPGWRK DD  ADD  ON DE AND     50
30  -      FILE CPGWRK     DEL  ON P1 AND NOT 50

31  -      FIELD PAGE. KEY   14
32  -                 SATZ1 78
```

Das Programm startet mit einem Bildschirmdialog mit Maske 1. Es kann der Schlüssel einer Datei eingegeben werden. Bei der Funktionstaste PF1 wird zum Programmabschnitt 'Datei pflegen' verzweigt, der beim Label A300 beginnt; ansonsten wird die Datei sequentiell gelesen und je 20 Sätze werden seitenweise angezeigt.

Beachte (Statement 15 und 17):

Vor dem Mapdialog mit Maske 2 (Anzeige einer Seite mit 20 Datensätzen) sollte der Zugriff auf die Datei, genauer die VSAM-Strings, freigegeben werden. Deshalb wird ein weiterer Satz gelesen und ein RANDOM für die Datei CPGWRK gegeben.

Im Detail und im Zusammenhang werden solche Programmiertechniken in unseren Seminaren erläutert.

Beispiel 21b: Programm zur Dateipflege, transaktionsorientiert 8110

Das folgende Programm ist aus Sicht des Benutzers identisch mit dem Programm in Teil a. Der Vorteil liegt darin, dass in der Zeit, in der der Benutzer einen Bildschirm bearbeitet, keine Task aktiv ist. Nach einem MAPO wird jeweils die Task verlassen. Erst beim erneuten Betätigen einer Programmfunktionstaste wird die Folgetask gestartet. Das leistet der Befehl EXITT (siehe Statement 57).

```
 1  - OPTIONS TITEL Dateipflege#taskorientiert ROOT PHASE TEST.

 2  - FILE CPGWRK
 3  - FILE TSQ.                      * Zwischenspeicherbereich

    - -D.
 4  -    I  3 0
 5  -    PAGE  20 * 78

    - -I.
 6  -    FILE CPGWRK DD
 7  -    FILE TSQ DD.                * Stelle 1:        KZ
                                     * Stellen 2 - 15:  KEY
    - -C.
 8  -    IF CPGMPF = 'CL'.           * Löschtaste = Programmende
 9  -      PURGE TSQ.                * Zwischenspeicher löschen
10  -      MAPO ENDE.
11  -    ELSE.                       * Programmausführung
12  -      1 READ TSQ.               * Zwischenspeicher lesen
13  -      EVALUATE.                 * Eine der folgenden Alternativen
    -        *-----------------------*
14  -        WHEN CPGMPF = 'P2' OR.  * wenn PF2 gedrückt oder
15  -        WHEN KZ = ' '.          * wenn erster Aufruf
16  -          KZ = '1'.
17  -          MAPO MASKE1.          * Key-Eingabemaske
    -        *-----------------------*
18  -        WHEN CPGMPF = 'DE' AND. * wenn Datenfreigabe gedrückt
19  -        WHEN KZ = '1'.          * und nicht erster Aufruf
20  -          MAP MASKE1.
21  -          DO WHILE I < 20 AND.  * Page noch nicht voll und
22  -             WHILE CPGFRC = ' '.  * End of File nicht erreicht
23  -               KEY READ CPGWRK
24  -               IF CPGFRC = 'EF'
25  -                 KZ = ' '
26  -               ELSE
27  -                 I = I + 1
28  -                 EDIT PAGE(I)
29  -               ENDIF
30  -          ENDDO
31  -          IF CPGFRC = '  '
32             KEY READ CPGWRK.     * Start-KEY für die Folge-Task
33  -          ENDIF
34  -          MAPO MASKE2.          * Sequentielle Anzeige
    -        *-----------------------*
```

```
    -                                 * Datensätze ändern/hinzufügen/
    -                                 *              löschen
35  -        WHEN KZ = '1'.           * Schritt 1: Anzeigen und ändern
    -                                 *            auf dem Bildschirm
36  -          MAP MASKE1.            *   KEY vom Bildschirm lesen
37  -          KEY CHAIN CPGWRK.      *   SATZ von Datei lesen
38  -          MAPO MASKE3.           *   SATZ anzeigen (und ändern)
39  -          KZ = '2'
    -        *------------------------*
40  -        WHEN KZ = '2'.           * Schritt 2: Ändern auf Datei
41  -          KEY CHAIN CPGWRK.      *   SATZ lesen
42  -          MAP  MASKE3.           *   Modifizierten SATZ vom Schirm
43  -          IF CPGMPF = 'P1'.      * PF1 gedrückt für Löschen
44  -            IF CPGFRC = '  '.    * Satz vorhanden
45  -              EXCPT WRK-LÖSCHEN
46  -            END
47  -          ELSE
48  -            IF CPGFRC = '  '.    * Satz vorhanden
49  -              EXCPT WRK-ÄNDERN
50  -            ELSE.                * Satz nicht vorhanden
51  -              EXCPT WRK-HINZUFÜGEN
52  -            ENDIF
53  -          ENDIF
54  -          MAPO MASKE1
55  -          MOVE '1' KZ.           * Nächster Schritt: Anzeigen
    -        *------------------------*
56  -      END-EVALUATE
57  -      EXCPT TS.                  * Zwischenspeicher beschreiben
58  -      EXITT 'TEST'.              * Task ruft sich selbst auf
59  -    ENDIF.                       * Abfrage CLEAR-Taste

    - -O.
60  -      FILE CPGWRK DD       WRK-ÄNDERN
61  -      FILE CPGWRK DD  ADD  WRK-HINZUFÜGEN
62  -      FILE CPGWRK     DEL  WRK-LÖSCHEN
63  -      FILE TSQ DD          TS
64  -      FIELD PAGE
65  -        KEY 14
66  -        SATZ1 78
```

Erläuterungen:

3 Die Daten werden im temporären Speicherbereich TSQ zwischengespeichert. Hier werden nur die Felder KZ und KEY an die Folgetask uebergeben. Als Satzlänge von TSQ wird deshalb im Data Dictionary 15 angegeben. Soll die Anwendung noch ausgebaut werden, sieht man eine (großzügige) Reserve bei der ersten Beschreibung vor, damit der Bereich problemlos während der Entwicklung vergrößert werden kann. Ein Q für Queueing ist heute Standard.

8 Bei transaktionsorientierter Programmierung ist der Programmierer für die Programmbeendigung selbst verantwortlich. Taskorientierte Programme sollten deshalb immer mit der Abfrage der Lösch-Taste (oder anderer Abbruchkriterien) beginnen.

9 Bei Programmende wird in der Regel der verwendete Zwischenspeicher gelöscht. Der Befehl PURGE löscht die gesamte Queue.

10 Am Ende der Verarbeitung steht eine Maske, in die (zumindest) die

TransId des Folgeprogramms eingegeben werden kann.

12 Bei 'normaler' Verarbeitung wird zunächst der Zwischenspeicher gelesen. Hier enthält er das Kennzeichen KZ, das aussagt, in welchem Verarbeitungsschritt sich das Programm befindet. Außerdem enthält er den zuletzt verarbeiteten KEY.

13 EVALUATE sorgt dafür, dass nur genau eine der folgenden Alternativen zur Ausführung kommt. Die einzelnen Alternativen werden mit WHEN eingeleitet (siehe 14, 18, 34, 39), END-EVALUATE (siehe 55) beendet den Befehl.

Beispiel 22a: Konvertieren von Feldinhalten mit CONVERT 8120

Im folgenden ist ein Programmauszug dargestellt, der die unterschiedlichen Möglichkeiten der Operation CONVERT beispielhaft erläutert:

	Statement	Feldinhalt F1	F2
1	- MOVE 'Test' TO F1	Test	
2	- CONVERT F1 LOW	test	
3	- CONVERT F1	TEST	
4	- CONVERT F1 INTO F2 HEX	TEST	E3C5E2E3
5	- FILL '*' F1	****	E3C5E2E3
6	- CONVERT F2 INTO F1 CHARACTER	TEST	E3C5E2E3
7	- CONVERT F2 INTO F2 CHAR	TEST	TESTE2E3
8	- MOVE 'F0F0' F1	F0F0	TESTE2E3
9	- CONVERT F1 INTO F2 CHAR	F0F0	00STE2E3

Zu 2: Grundsätzlich gilt: Zeichen, die nicht im Sinne der Operation konvertiert werden können, bleiben unverändert.

Im Beispiel: Es sollen alle Bytes des Feldes in Kleinbuchstaben geändert werden. Der einzige Großbuchstabe ist 'T', alle anderen Zeichen bleiben unverändert; wären z.B. Ziffern im Feld enthalten, dann blieben auch sie unverändert.

Bei der Konvertierung vom Hex- ins Characterformat muss der Programmierer aber sicherstellen, dass im Ursprungsfeld sinnvolle Werte, also Werte zwischen '00' und 'FF' stehen; sonstige Werte werden in diesem Fall zu X'00' konvertiert.

Zu 7 und 9: Grundsätzlich gilt: Das Ergebnisfeld wird vor der Operation nicht gelöscht.

Das Ergebnis der Konvertierung wird linksbündig im Ergebnisfeld abgestellt.

Beispiel 22b: Konvertieren von Zeitinformationen mit CONVERT 8125

CONVERT für Datumsfelder, Beispiele:

	Statement	ALFA6	ALFA8	ALFA10
1	- MOVE '220193' ALFA6	220193		
2	- CONVERT ALFA6 DATE	930122		
3	- CONVERT ALFA6 INTO ALFA8 DATUM	"	220193	
4	- MOVE '2001' ALFA8	"	22012001	
5	- CONVERT ALFA8 DATE	"	20010122	
6	- ALPHA8 = '23.02.93'	"	23.02.93	
7	- CONVERT ALFA8 DATE	"	93.02.23	
8	- CONVERT ALFA8 INTO ALFA10 DAT	"	"	23.02.93
9	- MOVE '1999' TO ALFA10	"	"	23.02.1999
10	- CONVERT ALFA10 DATE	"	"	1999.02.23

Zu 5: Hier sieht man, dass in manchen Fällen die Daten des Ursprungsfeldes nicht eindeutig zu interpretieren sind. Für diese Fälle steht die Servicefunktion YEAR zur Verfügung, die beim Konvertieren davon ausgeht, dass die Jahreszahl vorne (links) im Datenfeld steht.

Die Datumskonvertierung ist außerdem für sechs- bis achtstellige numerische Felder unterstützt. Eine Ausnahme von der Regel, dass immer alphanumerische Felder in alphanumerische bzw. numerische Felder in numerische konvertiert werden, ist das CONVERT mit dem Service UDAte:

```
11  - CONVERT ISO8 INTO ALFA10  UDATE-FORMAT
```

Hierbei wird das achtstellige numerische Feld ISO8 (19960109) konvertiert in ein Alphafeld und dort aufbereitet abgestellt, wie es dem UDATE-Format entspricht (09.01.1996).

CONVERT für Zeitfelder:

	Statement	NUM5	NUM5C	NUM70
11	- NUM5 = 10000	10000		
12	- CONVERT NUM5 INTO NUM5C SECONDS	10000	03600	
13	- NUM5 = 90000 + NUM5C	93600	"	
14	- CONVERT NUM5 INTO NUM70 SECS	"	"	0034560
15	- CONVERT NUM70 INTO NUM5C TIME	"	93600	

Zu 12: Die Konvertierung von 10000 (HMMSS, also 1:00:00, d.h 1 Stunde) in Sekunden ergibt 3.600.

Zu 14: Die Konvertierung von 9 Uhr und 36 Minuten ergibt 34.560 Sekunden.

Zu 15: Die Konvertierung von 34.560 Sekunden ergibt 9 Stunden und 36 Minuten.

Beispiel 23: Zwischnenspeichern in der Common Area (CPGCOM) 8130

CPGCOM bietet die Möglichkeit, über die Common Area Daten zwischenzuspeichern. Diese Form der Zwischenspeicherung wird insbesondere bei der Kommunikation mit anderen Programmiersprachen und Tools genutzt.

CPGCOM braucht vom Programmierer nicht als Feld definiert zu werden. Maximal können 4080 Bytes Daten übergeben werden. CPGCOM wird mit den Befehlen EDIT und SELECT verarbeitet.

```
- OPTIONS PHASE PROGCPG.
- -I.
-   FIELD CPGCOM.
- *    1  8  DATUM.              * achtstelliges Datum
-      9 18  WTAG.               * Wochentag im Klartext
-     19 20  KW.                 * Kalenderwoche
- -C.
-   DATUM = CPGDAT.             * Parameter füllen
-   EDIT CPGCOM.                * Common Area bereitstellen
-   EXPR  DRCOBOL.              * LINK in die COBOL-Datumroutine
-   SELECT CPGCOM.              * Ergebnisse aus CPGCOM lesen
-   IF WTAG = 'SAMSTAG'  OR
-   IF WTAG = 'SONNTAG'
-   :

- -O.
-   FIELD CPGCOM
-      DATUM   8
```

Ein CPG-Programm PROGCPG nutzt hier eine existierende Datum-Routine. Den Eingabe-Parameter erwartet die Routine in den Stellen 1 bis 8 der Common Area, das Ergebnis übergibt die Routine in den Stellen 9 bis 18 bzw. in den Stellen 19 und 20 der Common Area.

Feldaufbereitungen können auch nach Typen unterschieden werden. Wenn z.B. ein CPG-Programm mit mehreren COBOL-Programmen kommuniziert, so kann das Feld CPGCOM unterschiedliche Informationen enthalten. Mit dem Schlüsselwort TYPE können gezielte Aufbereitungen angesteuert werden

Beispiel:

```
- -C.  EDIT CPGCOM TYPE PROG1

- -O.  FIELD CPGCOM TYPE PROG1
-        ALFA20  20
-      FIELD CPGCOM TYPE PROG2
-        DATUMV   4 PACKED
-        TEXT    16
-        DATUMB  20 PACKED
```

Die Länge der Common Area kann auf einen gewünschten Wert gesetzt werden, z.B. wenn das aufgerufene Programm dies verlangt.

Dazu wird vor dem EXPR-Befehl ein numerisches Feld entsprechend gefüllt.

```
-   CAL  =  750.                * Common Area Länge: 750 Bytes
-   EDIT CPGCOM.                * Aufbereiten der Common Area
-   CAL EXPR PROGRM3.           * Unterprogrammaufruf
```

Beispiel 24: Sequentielle Verarbeitung von Satzarten (Segmente) 8140

Sowohl VSAM-Dateien als auch HL1-Strukturen können aus mehreren Satzarten aufgebaut sein.

Im folgenden Beispiel wird gezeigt, wie eine solche Datei bei der Eingabe verarbeitet werden kann.

```
OPTIONS PHASE TST000  TITEL Segment-Verarbeitung.
*
FILE AUFTRAG
*
INPUT DIVISION
   FILE AUFTRAG
       1  2  SA
       SEGMENT  KOPF  DD  TYPE 01  REFERENCE  AUFTRAG
       SEGMENT POSTEN DD  TYPE 02  REFERENCE  AUFTRAG
       SEGMENT  TEXT  DD  TYPE 03  REFERENCE  AUFTRAG
*
PROCEDURE DIVISION
   DO WHILE CPGFRC = '  '
      READ AUFTRAG
      IF CPGFRC >< 'EF'
        EVALUATE
          WHEN SA = '01'
            READI AUFTRAG  SEGMENT KOPF
          WHEN SA = '02'
            READI AUFTRAG  SEGMENT POSTEN
          WHEN SA = '03'
            READI AUFTRAG  SEGMENT TEXT
        END-EVALUATE
         :
         :  Verarbeitung
         :
      ENDIF
   ENDDO
```

Beachte:

- Die Segmente müssen in der Input Division unmittelbar im Anschluss an die zugehörige Datei beschrieben werden.

Beschreibung:

Beim sequentiellen Lesen wird zunächst immer nur die Satzart eingelesen. In Abhängigkeit von der gelesenen Satzart wird danach der gleiche Datensatz nochmals eingelesen.

Im Data Dictionary ist die Datei AUFTRAG mit den Satzarten 01, 02 und 03 beschrieben. Um diese unterschiedlichen Strukturen nun im Programm gezielt ansprechen zu können, erhalten sie Segmentnamen, im vorliegenden Fall KOPF, POSTEN und TEXT.

Über die Eintragung REFerence wird die Verbindung vom Segmentnamen

zur Satzart der Datei AUFTRAG im Data Dictionary hergestellt.

In der Procedure Division wird dann die gewünschte Satzart der Datei angesprochen, indem man bei der READI-Operation zusätzlich zum Dateinamen das Schlüsselwort SEGMent und einen Segmentnamen angibt.

Beispiel 25: Direkte Verarbeitung von Satzarten mit Segmenten 8145

Direkt-Verarbeitung von Segmenten ist nur mit dem Befehl CHAIN in Verbindung mit dem Schlüsselwort UPDate unterstützt.

Beachte: CHAIN für Update sperrt einen Satz bzw. ein ganzes Daten-CI bis zum nachfolgenden Update. Wird kein Update gemacht, so muss ein RANDOM programmiert werden.

```
OPTIONS PHASE TSTXXX  TITEL Segment-Verarbeitung.
*
FILE BESTELL
*
INPUT DIVISION
   FILE BESTELL
       1  2  SA
       SEGMENT SATZ1
           11 17 NFNR
*

PROCEDURE DIVISION
   :
   KEY CHAIN BESTELL UPDATE
   IF CPGFRC = '  '.                  * gefunden !
     EVALUATE
       WHEN SA = '01'.                * nächste freie Bestellnummer
         READI BESTELL  SEGMENT SATZ1
         NFNR = NFNR + 1.
         EXCPT ERSTER-SATZ
         NFNR = NFNR - 1
       WHEN SA = '02'.                * 'normaler Datensatz'
         EXCPT AENDERUNG
     END-EVALUATE
   ELSE
     EXCPT NEU
   ENDIF
   :
```

Beschreibung:

In einem transaktionsorientierten Programm stehen geänderte Daten zum Updaten oder Hinzufügen zur Datei bereit. Im ersten Satz der Datei steht (für den Fall der Neuanlage) die nächste freie Nummer für die Key-Aufbereitung.

Es ergibt sich folgende Problematik:

Für CHAIN steht zunächst nur eine Input-Beschreibung zur Verfügung. Es sind aber zwei unterschiedliche Funktionen auszuführen, je nachdem

mit welchem Key zugegriffen wird:
1. Update: Das CHAIN darf keine Dateidaten lesen, da diese die bereitstehenden Daten überschreiben würden.
2. Ermitteln der Satznummer für ein späteres Hinzufügen. Hier müssen Daten eingelesen werden, allerdings nur die Satznummer.

Ausweg: Einsatz von Segmenten !

Im Programm oben wird beim CHAIN grundsätzlich nur die Satzart eingelesen. Wird Satzart '01' gefunden, so wird die nächste freie Nummer mit READI eingelesen, dann hochgezählt und wieder upgedated. Beim READI wird immer ein Segment angegeben. Dieses ist in der Input Division unter der Datei beschrieben und enthält die Einlesevorschrift für weitere Daten des zuletzt gelesenen Satzes.

Bei Satzart '02' handelt es sich um einen gültigen Key,es werden keine Daten gelesen, sondern nur ein Update gemacht.

Beispiel 26: Logisch verknüpfte IF-Abfragen 8150

Für die logische Verknüpfung von IF- und WHEN-Abfragen gibt es verschiedene Syntaxregeln.

```
options root phase TSTxxx       * Source-Code in Groß-/Kleinschrift
        titel Demo#IF#AND#/#OR
        end
file stor
-d
  feld1 8
  feld2 8
  meld 24
-i
  file stor dd
-c
  if cpgmpf = 'CL' or.          * PF-Taste CLEAR
  if cpgmpf = 'PC' or.          * PF-Taste PF12
  if cpgmpf = 'QC'.             * PF-Taste PF24
     exhm aexit.                * allgemeine EXIT-Routine
  else
     1 read stor
     map BSP026
     if feld1 > '        ' and.  * Falsche Eingabe, wenn beide Felder
        feld2 > '        '   or. *   gefüllt sind   oder wenn Feld1
        feld1 = '        ' and.  *   leer ist und Feld2 mit einem '$'
        feld2 = '$'.             *   beginnt.
        meld = 'Falsche Eingabe !'
        cpgmcu = 'FELD1'.        * Cursorpositionierung
     else
        exhm absp26.             * Verarbeitung
     end
     mapo BSP026
     exitt 'BS26'
  endif
```

Regeln für logisch verknüpfte IF- oder WHEN-Abfragen:

- Zu einer Gruppe logisch verknüpfter Abfragen gehört (nur) ein ENDIF, WHEN hat kein zugehöriges END.

- Ab der zweiten Abfrage kann das IF oder WHEN codiert werden (wie oben bei der CPGMPF-Abfrage) oder weggelassen werden (wie bei der Abfrage von FELD1 und FELD2).

- Klammern ist nicht möglich. Eindeutigkeit der Abfrage ist gegeben durch die Regel 'AND bindet stärker als OR' (vergleichbar der bekannten Regel 'Punktrechnung geht vor Strichrechnung').

  Eine dritte Möglichkeit der Codierung, die im Beispiel nicht angewandt ist, ist das Weglassen des Operationscodes in Verbindung mit der Möglichkeit, mehrere Statements in einer Zeile zu codieren. Der Punkt zwischen den Statements entfällt hier, da es sich um eine logische Einheit handelt. (Ausnahme !)

  ```
  WHEN A = 0 OR B = 0 OR C = 0
  ```

Beispiel 27: Groß-/Kleinschreibung in taskorientierter Anwendung 8160

Das folgende Beispiel soll zeigen, wie die Übersetzung in Großbuchstaben deaktiviert, der ursprüngliche UCTRN-Status gesichert und am Programmende wieder hergestellt wird. (UCTRN steht für Upper Case Translation und ist ein Parameter der Terminal Control Table des CICS)

```
 1  OPTIONS ROOT PHASE TST031.
 2  FILE STOR UPD QUE FIX 1 STORAGE. * Zwischenspeichern des  UCTRAN-
                                     * status
    -I.
 3     FILE KANAL HS
 4        1  1    STATUS.
 5     FILE STOR DD
    -C.
 6     1 READ STOR.                  * UCTRN-Status bei Progr.-Start
 7     IF CPGMPF = 'CL'.             * Lösch-Taste
 8       EXHM PENDE KANAL T.         * UCTRN zurücksetzen
 9       PURGE STOR.                 * Zwischenspeicher löschen
10       MAPO ENDE
11     ELSE
12       IF CPGFRC = 'EF'.           * Wenn erster Programmaufruf
13         EXHM PSTART KANAL.        * UCTRN-Status ermitteln/retten
14       ENDIF
15       MAP TEST LOW.               * Service 'LOW' damit der MAP-
         :                           * Befehl nicht in Großbuchstaben
         :                           * übersetzt
17       MAPO TEST
18       UPDAT STOR.                 * UCTRN-Status zwischenspeichern
19       EXITT 'TT31'
20     ENDIF
```

Erläuterungen:

Zu 8: Service-Funktion 'T' wie transaktionsorientiert beim Operationscode EXHM:

Wird in einem dialogorientierten Programm die Lösch-Taste betätigt, so wird automatisch zum Programmende verzweigt. In transaktionsorientierten Programmen wird der Weg zum Programmende vom Programmierer bestimmt, die automatische Programmbeendung ist deshalb nicht sinnvoll.

Die Operation EXHM prüft beim Rücksprung ins rufende Programm, ob die Lösch-Taste gedrückt ist und verzweigt dann ans Ende des Programms. Diese Automatik m u s s in transaktionsorientierten Programmen durch die Servicefunktion 'Task' ausgeschaltet werwerden.

Zu 15: Nicht nur CICS kennt die Übersetzung in Großbuchstaben beim Bildschirm-Input. Auch CPG übersetzt standardmäßig in Großbuchstaben. Diese Übersetzung vom CPG wird mit der Servicefunktion LOW beim MAP-Befehl ausgeschaltet.

Das folgende HL1-Modul erhält über die Operation COMRG den UCTRAN-Status des CICS.

```
OPTIONS PHASE PSTART.
-D.
        STATUS  1.                    * Datenkanal
                                      *_____________________________*
        SYSINF  32.                   * COMRG-Bereich, muss 32-stellig
        ORG SYSINF.                   *                definiert sein
           DUMMY1  21.                *
           UCTALT  1.                 * Stelle 22, UCTRN bis CPG 2.0
           DUMMY2  4.                 *
           UCTR  1.                   * Stelle 27, erweitertes UCTRN
           ORG.                       * Ende der Redefinition
-C.
        COMRG SYSINF.                 * Ermittelt den UCTRAN-Status
        UCTRN OFF.                    * Grundsätzlich: UCTRN OFF
                                      * _____________________________*
        IF UCTR >< UCTALT AND.        * Stelle 27 beim COMRG wird erst
        IF UCTR >< 'T'.               *   ab CICS 2.2 gefüllt. Deshalb
          UCTR = UCTALT               *   sollte sicherheitshalber die
        ENDIF                         *   Stelle 22  noch  berücksich-
                                      *   tigt werden.
                                      * _____________________________*
        STATUS = UCTR.                * UCTRAN-Status an das Kanalfeld
                                      * STATUS übergeben
```

Erläuterungen:

Seit CPG-Release 2.1 steht der UCTRN-Status an zwei Stellen im COMRG-Bereich: In Stelle 22 (U oder N) und in Stelle 27 (U, N oder T). Die Eintragung T steht für UCTRN Transaction. Stelle 27 wird vom CPG gefüllt, falls ein CICS Release 2.2 oder höher im Einsatz ist.

Das folgende HL1-Modul setzt den UCTRN-Status auf den Stand vor Aufruf des Programms zurück.

```
OPTIONS PHASE PENDE.
-D.
        STATUS  1.
-C.
        IF STATUS = 'U'.              * Wenn der Status 'U' war, dann
          UCTRN ON.                   *   UCTRN wieder an
        ELSE.
          IF STATUS = 'T'.            * Wenn der Status 'T' war, dann
            UCTRN ON TRANSACTION.     *   UCTRN TRANSACTION wieder an
          ENDIF
        ENDIF
```

Beispiel 45: Massen-Insert im CICS für VSAM-Dateien 8190

Massen-Insert ist eine Funktion im CICS zum schnellen Hinzufügen von Sätzen in ESDS-, RRDS- und KSDS-Dateien.

Besonderheiten:

- Für KSDS-Dateien müssen die hinzuzufügenden Sätze nach ihrem Schlüssel aufsteigend sortiert sein.
- Während des Massen-Inserts sind keine anderen Dateioperationen möglich !
- Die Verarbeitungsart Massen-Insert wird **„eingeschaltet“**, wenn in der Files Division die Verarbeitungsart Output gesetzt ist und Sätze mit EXCPT und dem Eintrag ADD in der Output Division ausgegeben werden.

  Die Verarbeitungsart Massen-Insert wird **„ausgeschaltet“**, wenn die Transaktion beendet wird oder wenn ein RANDOM für die Ausgabedatei ausgeführt wird.

Für die Programmierung bedeutet das, dass neben dem Massen-Insert in einem Programm oder Modul keine weitere Dateiverarbeitung möglich ist.

Wird ein Modul geschrieben, das aus anderen Programmen aufgerufen wird, dann sollte es sicherheitshalber die Massen-Insert-Verarbeitung mit einem RANDOM beenden.

```
OPTIONS PHASE SAMPLE45.
-   - *           TEST CPGESD ADD ONLINE MASSINSERT                *
    - *-------------------------------------------------------------*
    - FILE CPGESD OUT  VAR 4000 200 04 RBA ESDS.
    - *-------------------------------------------------------------*
    - -D.
    -       RECORD        80.
    -       COUNT          7 0.
    -       I              5 0.
    -       J              5 0.
    -       CPGVRL         5 0.
    -       STIME          7 0.
    - *-------------------------------------------------------------*
    - -C.
    -       FILL 'B' TO RECORD.      * SET RECORD TO 'B'
    -       CPGVRL = 200.            * SET RECORD LENGTH TO 200
    -       STIME = CPGTIM.          * SET START TIME TO STIME
    -       DO 10 WITH J.            * LOOP FOR 20
    -          TIME.                 * GET CURRENT TIME
    -          MAPO TT2501.          * TERMINAL DISPLAY EVERY 1000 REC
    -          DO 1000 WITH I.       * LOOP FOR 1000
    -             EXCPT SATZ.        * WRITE ESD RECORD AS MASS INSERT
    -          ENDDO.                * END LOOP 1000
    -       ENDDO.                   * END LOOP 10
    -       STIME = CPGTIM - STIME.  * GET DURATION
    -       MAPO TT2501.             * DISPLAY DURATION
    -       RANDOM CPGESD.           * RELEASE MASSINSERT
    - *-------------------------------------------------------------*
    - -O.
    -   FILE  CPGESD ADD SATZ.
    -       RECORD    80.
```

Beispiel B01 : Leser, Drucker und Überlaufsteuerung.

```
 1  - OPTIONS  BATCH  PHASE BSPB01.

 2  - FILE READER  INPUT  FIX  80   READER
 3  - FILE PRINTER OUTPUT FIX 132   PRINTER

 4  - FORMS PRINTER  LENGTH 72  LINE 1 CHANNEL 1  LINE 66 CHANNEL 12

    - -I
 5  -     FILE READER
 6  -        1 80 SATZ

    - -C
 7  -     EXCPT  KOPF
 8  -     DO UNTIL CPGFRC >< '  '
 9  -       READ READER
10  -       IF CPGFRC >< 'EF'
11  -         IF CONDITION OF.        * Overflow
11  -           EXCPT  SEITENWECHSEL
11  -         ENDIF
12  -         EXCPT  DATEN
    -       ENDIF
13  -     ENDDO

    - -O
15  - FILE PRINTER  SPACE # 2  SKIP 01  KOPF
16  -               24 'Erste Seite'
17  -     UDATE     70
18  -     UTIME     80
19  - FILE PRINTER  SPACE # 2  SKIP 01  SEITENWECHSEL
20  -               24 'Folgeseite '
21  - FILE PRINTER  SPACE # 1           DATEN
22  -     SATZ      80
```

Erläuterungen:

1 Options BATch muss für Batchprogramme angegeben werden.

2 Es wird eine Datei 'READER' als Eingabedatei (INPUT) definiert. Die Satzlänge ist fest (FIX) 80 Bytes. Die Einheit ist READER (SYSIPT).

3 Es wird eine Druckerdatei 'PRINTER' als Ausgabedatei (OUTPUT) definiert. Die Satzlänge ist fest (FIX) 132 Bytes. Die Einheit ist 'PRINTER' (SYSLST).

4 Die Formularlänge wird mit 72 Zeilen definiert. Die erste Zeile wird mit Kanal 01 und Zeile 66 mit Kanal 12 definiert.(Kanal 12 beschreibt die Zeile, nach der der Überlaufschalter 'OF' gesetzt wird).

Auf die FORMS-Einträge kann vollständig verzichtet werden. In diesem Fall gelten die Defaults: Kanal 01 = Zeile 6, Kanal 12 = Zeile 66.

5 In der Input Division wird die Lesevorschrift für die Datei READER beschrieben.

6 Vom READER werden die Stellen 1 bis 80 in das Feld SATZ gelesen.

7 Ausführung der Ausgabe. Es werden alle unverriegelten Satzbestimmungen ausgeführt und die Ausgaben, deren Satzbestimmung den Namen KOPF hat. Da alle Ausgaben verriegelt sind, wirkt dies wie ein direktes Ansteuern der Ausgabe mit dem Namen KOPF.

8 Beginn einer Schleife, die beendet ist, wenn der File Return Code nicht mehr Blank ist (bei End of File).

9 Die Eingabedatei 'READER' wird gelesen.

10 'End of File' wird als File Return-Code abgefragt (EF in CPGFRC).

11 Der Überlaufschalter wird abgefragt; ist er gesetzt, wird die Ausgabe SEITENWECHSEL ausgeführt.

12 Ausführung der Ausgabe mit dem Namen DATEN.

13 Es wird wieder zum Anfang der Schleife (Statement 8) verzweigt.

15 Diese Ausgabe wird mit EXCPT KOPF angesteuert (siehe 7).

Vor dem Drucken wird auf Kanal 01 vorgesetzt, d.h. auf den Beginn der nächsten Seite (SKIP 01). Vor dem Drucken findet kein Zeilentransport statt, nach dem Drucken wird um 2 Zeilen vorgeschoben (SPACE # 2).

16 Ausgabe der Überschrift, Datum und Uhrzeit.

19 Vor dem Drucken nach Kanal 01, nach dem Drucken 2 Zeilentransporte.

20 Ausgabe einer Konstanten.

21 Ein Zeilentransport nach Ausgabe des Feldes SATZ, wenn EXCPT DATEN
22 ausgeführt wird.

Beispiel B02 : Programmkatalog CPGWRK (siehe auch nächste Seite !)

```
 1  - OPTIONS  BATCH  TITEL Programmkatalog  PHASE BSPB02.

 2  - FILE PRINTER
 3  - FILE CPGWRK  INPUT

    - -D.
 4  - KEY    14
 5  - STRICH 80

    - -I.
 7  - FILE CPGWRK.
 8  -     1   2   SA
 9  -     3  10   PNAME
10  -    11  14   TRANID
11  -    15  24   PROT
12  -    25  29 0 TWA
13  -    30  49   TEXT
14  -    50  52   PKZ

    - -C.
13  -    FILL '-' STRICH
14  -    EXCPT  01
15  -    MOVEL '01 ' TO KEY
16  -    KEY  SETLL  CPGWRK
17  -    DO UNTIL CPGFRC = 'EF'
18  -       KEY  READ  CPGWRK
19  -       IF CPGFRC >< 'EF'  AND
20  -       IF SA <= KEY
21  -          EXCPT 02
22  -       ENDIF
23  -    ENDDO

    - -O.
24  - FILE PRINTER  SPACE # 2  SKIP 01  ON 01
25  -                                   15 'PROGRAMMKATALOG'
26  -                          UDATE     70
27  -                          UTIME     80
28  - FILE PRINTER  SPACE # 2           ON 01
29  -       STRICH  80
30  - FILE PRINTER  SPACE # 2           ON 01
31  -               22 'PROGRAMM   TRID   TEXT'
32  -               67 'PKZ     TWA   PROTECTION'
33  - FILE PRINTER  SPACE # 1           ON 02
34  -       PNAME    8
35  -       TRANID  15
36  -       TEXT    38
37  -       PKZ     46
38  -       TWA     54  EDITCODE J
39  -       PROT    67
```

Erläuterungen:

In diesem Beispiel wird mit einem Key in der Datei 'CPGWRK' positioniert und eine Liste (Programmkatalog) auf dem Drucker ausgegeben. Bei Ende der Datei (Schalter EF) oder einer neuen Satzart wird das Programm beendet.

Beispiel B02, zeitgemäß codiert:

Durch Einsatz der angebotenen Werkzeuge lässt sich das Coding wesentlich verkürzen. Der Druckoutput wird programmextern entwickelt und gepflegt, Datei-Ein- und -Ausgabestrukturen werden zur Zeit der Umwandlung dem Data Dictionary entnommen.

Es bleibt dann zu codieren:

```
 1  - OPTIONS  BATCH  TITEL Programmkatalog  PHASE BSPB02.


 2  - FILE PRINTER
 3  - FILE CPGWRK  INPUT

    - -D.
 4  -    KEY   14

    - -I.
 5  -    FILE CPGWRK DD

    - -C.
 6  -    LIST BSPB02 SECTION KOPF
 7  -    KEY = '01 '
 8  -    DO UNTIL CPGFRC = 'EF'
 9  -       KEY READ CPGWRK
10  -       IF CPGFRC >< 'EF' AND SA <= KEY
11  -          LIST BSPB02 SECTION DATEN
12  -       ENDIF
13  -    ENDDO
```

Beispiel B03 : Sätze vom Kartenleser in eine KSDS-Datei laden.

```
- OPTIONS BATCH TITEL Load#CPGKSD PHASE BSPB03.

- FILES READER PRINTER
- FILE CPGKSD OUTPUT

- -D
-    STRICH  80

- -I
-    FILE READER
-       1 20 KEY
-      21 80 DATEN

- -C
-    FILL '-' STRICH
-    EXCPT KOPF
-    DO UNTIL CPGFRC = 'EF'
-       READ READER
-       IF CPGFRC >< 'EF'
-          EXCPT SÄTZE
-       ENDIF
-    ENDDO

- -O
- FILE PRINTER  SPACE # 2  SKIP 01   KOPF
-             18 'CPGKSD NEUE SÄTZE'
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 2            KOPF
-      STRICH 80
- FILE PRINTER  SPACE # 2            KOPF
-              3 'KEY'
-             26 'DATEN'
- FILE PRINTER  SPACE # 1            SÄTZE
-      KEY    20
-      DATEN  81
- FILE CPGKSD   ADD                  SÄTZE
-      DATEN  80
-      KEY    20
```

Erläuterungen:

Im obenstehenden Beispiel werden Karten von einem Kartenleser gelesen und in die KSDS-Datei CPGKSD übertragen. Die gelesenen Sätze werden mit 'ADD' sequentiell hinzugefügt, d. h. wenn bereits Sätze in der Datei existieren, werden neue Sätze in die Datei eingefügt.

FILE CPGKSD OUTPUT in der Files Division kennzeichnet die Datei als Ausgabedatei. Die Ausgabe erfolgt sequentiell. Beim sequentiellen Hinzufügen müssen die Sätze in aufsteigender Schlüsselfolge vorliegen. Beim Laden der Datei wird gleichzeitig auf der Datei 'PRINTER' ein Protokoll gedruckt.

Beispiel B04 : Sätze von der CPGWRK in eine KSDS-Datei kopieren.

```
- OPTIONS  PHASE BSPB04
-          TITEL Copy#KSDS#to#KSDS
-          BATCH
-          END

- FILES READER PRINTER
- FILE CPGWRK  INPUT
- FILE CPGKSD  OUTPUT

- -D
-    EC      2
-    STRICH 80

- -I
-    FILE READER
-       1   2 SA
-       1  14 KEYVL
-    FILE CPGWRK
-       1  14 KEY
-      15 100 SATZ

- -C
-    FILL '-' STRICH
-    DO WHILE EC >< 'EF'
-       EC = '  '
-       READ READER
-       IF CONDITION EOF
-          EC = 'EF'
-       ELSE
-          EXCPT  KOPF
-          KEYVL SETLL CPGWRK
-          DO WHILE EC = '  '
-             KEYVL READ CPGWRK
-             IF CONDITION EOF
-                EC = 'EF'
-             ELSE
-                IF KEY = SA
-                   EXCPT  DATEN
-                ELSE
-                   EC = 'SA'
-                ENDIF
-             ENDIF
-          ENDDO
-       ENDIF
-    ENDDO

- -O
- FILE PRINTER  SPACE # 2  SKIP 01   KOPF
-             15 'SÄTZE KOPIEREN'
-             38 'VON CPGWRK NACH CPGKSD'
-             50 'SATZART'
-      SA     53
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 2            KOPF
-      STRICH 80
- FILE PRINTER  SPACE # 2            KOPF
-              3 'KEY'
```

```
-             25 'SATZ'
- FILE PRINTER  SPACE # 1            DATEN
-      KEY    14
-      SATZ  107
- FILE CPGKSD   ADD                  DATEN
-      KEY    14
-      SATZ  106
```

Erläuterungen:

Im obenstehenden Beispiel werden Sätze aus der 'CPGWRK' sequentiell in eine KSDS-Datei kopiert. Die Sätze werden mit 'ADD' in die KSDS-Datei übertragen.

Über die Datei 'READER' werden Vorlaufkarten eingelesen, die die zu kopierenden Satzarten angeben.

Auf der Datei 'PRINTER' wird ein Protokoll ausgedruckt.

Beispiel B05 : Sätze in einer KSDS-Datei direkt updaten.

```
    - OPTIONS  BATCH  TITEL Direktes#Update  PHASE BSPB05.

    - FILES READER PRINTER CPGKSD

    - -I
    -    FILE READER
    -       1  20 KEY
    -      21  21 UPDKZ
    -    FILE CPGKSD
    -      21  39 NAME

    - -C
    -     EXCPT KOPF
    -     DO UNTIL CPGFRC = 'EF'
    -        READ READER
    -        IF CPGFRC >< 'EF'
**  -           KEY CHAIN CPGKSD  99
    -           EXCPT DATEN
    -        ENDIF
    -     ENDDO

    - -O
    - FILE PRINTER  SPACE # 2  SKIP 01          KOPF
    -             24 'Sätze ändern in CPGKSD'
    -      UDATE  70
    -      UTIME  80
    - FILE PRINTER  SPACE # 2                   KOPF
    -              3 'KEY'
    -             25 'NAME'
    -             42 'X'
    - FILE PRINTER  SPACE # 1                   DATEN
    -      KEY    20
    -      NAME   40
    -      UPDKZ  42
    -   ON 99  #  43 '*** nicht gefunden ***'
    - FILE CPGKSD   ON NOT 99                   DATEN
    -      NAME   59
    -      UPDKZ 106
```

Erläuterungen:

In diesem Beispiel werden Sätze, die von einem Kartenleser eingelesen wurden, direkt in einer KSDS-Datei upgedated. Ein Protokoll wird auf dem Drucker ausgedruckt.

** Beachte: Ein Schalter beim CHAIN muss angegeben werden, wenn das interne Feld CPGFRC für den File Return-Code im Programm nicht abgefragt wird.

Beispiel B06 : Sätze in einer KSDS-Datei sequentiell updaten.

```
- OPTIONS  BATCH  TITEL sequentielles#Updat  PHASE BSPB06.

- FILE LISTE OUTPUT FIX 132 PRINTER
- FILE CPGKSD

- -I. FILE CPGKSD
-     1  20  KEY
-     1  40  SATZ

- -C.
-     EXCPT KOPF
-     MOVEL '11' TO KEY
-     DO WHILE KEY = '11'
-        KEY READ CPGKSD
-        IF KEY = 11
-           EXCPT SEQUPD
-        ENDIF
-     ENDDO

- -O.
- FILE LISTE    SPACE # 2  SKIP 01  KOPF
-             16 'SÄTZE IN CPGKSD'
-             36 'SEQUENTIELL UPDATEN'
-      UDATE  70
-      UTIME  80

- FILE LISTE    SPACE # 1           SEQUPD
-      SATZ   40
-             40 '*'

- FILE CPGKSD                       SEQUPD
-             40 '*'
```

Erläuterungen:

In diesem Beispiel werden Sätze sequentiell in einer KSDS - Datei upgedated und gleichzeitig als Liste auf dem Drucker ausgegeben.

Es werden nur die Sätze upgedated, die den Schlüssel '11' haben.

Beispiel B07 : Sätze in einer KSDS-Datei sequentiell löschen.

```
- OPTIONS  BATCH  TITEL sequentiell#löschen PHASE BSPB07.

- FILES READER PRINTER CPGKSD

- -D
-    EOF  2
- -I
-    FILE READER
-       1   2  SA
-    FILE CPGKSD
-       1  20  KEY
-      21 100  SATZ
- -C
-    DO WHILE EOF >< 'ER'.                  * EOF READER
-       READ READER
-       IF CPGFRC = 'EF'
-          EOF = 'ER'
-       ELSE
-          EXCPT KOPF
-          DO WHILE EOF >< 'ED'.            * EOF CPGKSD  oder
-             SA READ CPGKSD.               * Ende der Satzart
-             IF CPGFRC = 'EF'
-                RANDOM CPGKSD
-                EOF = 'ED'
-             ELSE
-                IF KEY = SA
-                   EXCPT LÖSCHEN
-                ELSE
-                   RANDOM CPGKSD
-                   EOF = 'ED'
-                ENDIF
-             ENDIF
-          ENDDO
-          FILL ' ' EOF
-       ENDIF
-    ENDDO
- -O
- FILE PRINTER  SPACE # 2  SKIP 01          KOPF
-             24 'Delete Records in CPGKSD'
-             34 'SATZART'
-      SA     37
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 2                   KOPF
-              3 'KEY'
-             25 'SATZ'
- FILE PRINTER  SPACE # 1                   LÖSCHEN
-      KEY    20
-      SATZ  101
- FILE CPGKSD   DEL                         LÖSCHEN
```

Erläuterungen:

In diesem Beispiel werden Sätze sequentiell in einer KSDS-Datei gelöscht und ein Protokoll wird auf dem Drucker ausgegeben.

Beispiel B08 : Sätze in einer KSDS-Datei direkt löschen.

```
-   OPTIONS  BATCH  TITEL direkt#löschen PHASE BSPB08.

-   FILES READER PRINTER CPGKSD

- -D
-   EOF  2

- -I
-   FILE READER
-               1  20 KEY
-   FILE CPGKSD
-              21  39 NAME

- -C
-    EXCPT KOPF
-    DO WHILE CPGFRC = '  '
-       READ READER
-       IF CPGFRC = '  '
-          KEY CHAIN CPGKSD  99
-          EXCPT LÖSCHEN
-       ENDIF
-    ENDDO

- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze in der CPGK'
-             34 'SD direkt löschen'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 2           KOPF
-              3 'KEY'
-             25 'NAME'

- FILE PRINTER  SPACE # 1           LÖSCHEN
-      KEY    20
-      NAME   40
-    ON 99  # 43 '*** nicht gefunden ***'

- FILE CPGKSD DELETE      ON NOT 99 LÖSCHEN
```

Erläuterungen:

Es werden Sätze in einer KSDS-Datei direkt gelöscht. Ein Protokoll wird auf dem Drucker ausgegeben.

Beispiel B09 : Programmkatalog rückwärts auflisten.

```
- OPTIONS  BATCH TITEL Katalog#rückwärts PHASE BSPB09.
- FILES PRINTER CPGWRK
- -D. OC       2
-     STRICH  80
- -I. FILE CPGWRK.    * DD - Felder dem Data Dictionary entnehmen
-                     *
-                                          1   2 SA
-                                          3  10 PNAME
-                                         11  14 TRANID
-                                         15  24 PROT
-                                         25  290TWA
-                                         30  49 TEXT
-                                         50  52 PKZ
- -C. FILL '-' STRICH
-     EXCPT KOPF
-     '02' READB CPGWRK.          * Der Satz muss vorhanden sein !
-     DO WHILE OC = '  '
-        '01' READ-BACK CPGWRK.   * Letzter 01-Satz wird gelesen
-         IF CPGFRC = 'EF'
-            OC = 'EF'
-         ELSE
-            IF SA = '01'
-               EXCPT  SATZ
-            ELSE
-               OC = 'ED'
-            ENDIF
-         ENDIF
-      ENDDO
- -O
- FILE PRINTER  SPACE # 1  SKIP 01  KOPF
-             15 'Programmkatalog'
-             28 'rückwärts'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 2           KOPF
-      STRICH 80

- FILE PRINTER  SPACE # 2           KOPF
-             22 'Programm   TRID   Text'
-             67 'PKZ     TWA   Protection'

- FILE PRINTER  SPACE # 1           SATZ
-      PNAME   8
-      TRANID 15
-      TEXT   38
-      PKZ    46
-      TWA    54  EDIT J
-      PROT   67
```

Erläuterungen:

Es wird in einem Programmkatalog rückwärts gelesen (READB) und ein Protokoll auf dem Drucker ausgegeben.

Beispiel B10 : Sätze vom Kartenleser in eine ESDS-Datei kopieren.

```
- OPTIONS  BATCH  TITEL Copy#Leser#->#ESDS  PHASE BSPB10.

- FILES READER CPGESD

- -I. FILE READER.  1 80 SATZ

- -C. DO LOOP
-        READ READER
-        ON EOF  BREAK
-        EXCPT
-     ENDDO

- -O. FILE CPGESD  ADD.  SATZ 80
```

Erläuterungen:

In diesem Beispiel werden Sätze von einem Kartenleser in eine ESDS-Datei kopiert. Die Sätze werden mit 'ADD' zur ESDS-Datei hinzugefügt.

Bestehen bereits Sätze in der ESDS-Datei, so werden die neuen Sätze an das Ende der Datei angefügt.

BeispieL B11 : Drucken einer ESDS-Datei.

```
- OPTIONS  BATCH  TITEL ESDS#drucken  PHASE BSPB11.

- FILES CPGESD PRINTER

- -D. COUNT  5 0
-     KEY    4
-     OC     2

- -I. FILE CPGESD
-        1  80  SATZ

- -C. EXCPT KOPF
-     FILL  X'00'  KEY
-     DO WHILE CPGFRC = '  '
-        KEY READ CPGESD
-        IF CPGFRC = 'EF'
-           EXCPT SUMME
-        ELSE
-           COUNT = COUNT + 1
-           EXCPT SÄTZE
-        ENDIF
-     ENDDO
- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGESD'
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80
- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      COUNT  23 EDITCODE Z
```

Erläuterungen:

Im obigen Beispiel wird eine ESDS-Datei gelesen und auf dem Drukker ausgegeben.

Die Verarbeitung beginnt mit dem 1. Satz der Datei.
(FILL X'00' nach KEY).

Beispiel B12 : Sequentielles Update einer ESDS-Datei.

```
- OPTIONS  BATCH  TITEL ESDS#seq#update  PHASE BSPB12.

- FILES CPGESD PRINTER

- -D. COUNT  5 0
-     KEY  4
-     OC  2

- -I. FILE CPGESD
-        1  80  SATZ

- -C. EXCPT KOPF
-     FILL  X'00'  KEY
-     DO WHILE CPGFRC = '  '
-        KEY READ CPGESD
-        IF CPGFRC = 'EF'
-           EXCPT SUMME
-        ELSE
-           COUNT = COUNT + 1
-           EXCPT SÄTZE
-        ENDIF
-     ENDDO

- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGESD'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80

- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      COUNT  23 EDITCODE Z

- FILE CPGESD                       SÄTZE
-             40 '<--- SEQ.UPD --->'
```

Erläuterungen:

In obigem Beispiel werden Sätze sequentiell in einer ESDS-Datei upgedated und gleichzeitig als Liste auf dem Drucker ausgegeben.

Als Einheit der Datei muss im Data Dictionary 'ESDS' eingetragen sein.

Die Verarbeitung beginnt mit dem 1. Satz der Datei.

Beispiel B13 : Direkte und sequentielle Verarbeitung einer ESDS-Datei.

```
- OPTIONS  BATCH  TITEL ESDS-Verarbeitung  PHASE BSPB13.
- FILES CPGESD PRINTER
- -D
-    COUNT  5 0.                  * Zähler
-    KEY  4.                      * Schlüssel (RBA)
-    KEYS  99 * 4.                * Schlüssel-(RBA)-Feldgruppe
-    OC  2.                       * Operationscode
-    X  3 0.                      * Feldgruppenindex
- -I
-    FILE CPGESD
-       1  80   SATZ
- -C
-    EXCPT  KOPF
-    FILL  X'00'  KEY
-    DO UNTIL X =  99  OR
-       UNTIL CPGFRC = 'EF'
-       KEY READ CPGESD
-       IF CPGFRC >< 'EF'
-         X = X + 1
-         KEYS(X) = CPGK01
-         EXCPT SÄTZE
-       ENDIF
-    ENDDO
-    EXCPT SUMME
-    RANDOM CPGESD
-    DO 5 TIMES WITH X
-       KEY = KEYS(X)
-       KEY CHAIN CPGESD   11
-       EXCPT UPDESD
-    END
- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGESD'
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80
-             93 'sequentiell'
- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      X      23 EDIT  Z
- FILE PRINTER  SPACE # 1           UPDESD
-      SATZ   80
-             93 'direkt '
-    ON 11 # 110 'nicht gefunden'
- FILE CPGESD                       UPDESD
-             20 '<<< direkt >>> '
```

Erläuterungen:

Hier wird eine ESDS-Datei sowohl direkt ('CHAIN'), als auch sequentiell ('READ') verarbeitet. Durch den Befehl 'RANDOM' wird von der sequentiellen Verarbeitung umgeschaltet auf die direkte Verarbeitung, gleichzeitig wird ein Protokoll auf der Datei 'PRINTER' gedruckt.

Der direkte Zugriff darf nur mit einer gültigen relativen Byte-Adresse (RBA) erfolgen. In diesem Beispiel wurden die relativen

Byte-Adressen beim sequentiellen Lesen in der Feldgruppe KEYS gespeichert.

Beispiel B14 : Drucken einer RRDS-Datei.

```
- OPTIONS  BATCH  TITEL RRDS#drucken PHASE BSPB14.

- FILES CPGRRT PRINTER

- -D
-   COUNT  5 0
-   KEYN  9 0
-   OC  2

- -I
-   FILE CPGRRT
-      1  80  SATZ

- -C
-   EXCPT KOPF
-   KEYN = 1
-   DO WHILE CPGFRC = '  '
-      KEYN READ CPGRRT
-      IF CPGFRC = '  '
-         COUNT = COUNT + 1
-         EXCPT SÄTZE
-      ENDIF
-   ENDDO
-   EXCPT SUMME

- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGRRT'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80

- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      COUNT  23
```

Erläuterungen:

In obigem Beispiel wird eine RRDS-Datei gelesen und auf dem Drukker ausgegeben.

Die Verarbeitung beginnt mit dem 1. Satz der Datei.

Beispiel B15 : RRDS-Datei direkt verarbeiten.

```
- OPTIONS  BATCH  TITEL RRDS#direkt  PHASE BSPB15.

- FILES READER PRINTER CPGRRT

- DATA DIVISION
-    COUNT  5 0.                  * Zähler
-    INFO  14.                    * Info für Duplicate Record
-    SERVICE  7.                  * Textfeld für Service

- INPUT DIVISION
-    FILE READER
-       1   5 0  X.               * Relative Satznummer
-       6   6    OC.              * Operationscode
-       1  20    FELD
-    FILE CPGRRT
-       1  80    SATZ

- PROCEDURE DIVISION
-    EXCPT KOPF
-    DO UNTIL OC = 'E'.           * Bis End of File
-       READ READER
-       IF CPGFRC = 'EF'
-          MOVE 'E' TO OC.        * Abbruchkriterium der Schleife
-       ELSE
-          X  CHAIN CPGRRT  11
-          COUNT = COUNT + 1
-          MOVEL X  TO  SATZ
-          MOVE  FELD  TO  SATZ
-          EVALUATE
-            WHEN OC = '*'.       *------ Service: Ändern -----*
-              SERVICE = 'UPDATE'
-              EXCPT  RRTUPD
-            WHEN OC = '+'.       *------ Service: Hinzufügen -*
-              SERVICE = 'WRITE '
-              EXCPT RRTADD
-            WHEN OC = '-'.       *------ Service: Löschen ----*
-              SERVICE = 'DELETE'
-              EXCPT RRTDEL
-          END-EVALUATE
-          IF CPGFRC = 'D'
-             INFO = 'doppelter Satz'
-          ENDIF
-          EXCPT  SÄTZE
-       ENDIF
-    ENDDO
-    EXCPT SUMME

- -O
- FILE PRINTER  SPACE # 2  SKIP 01          KOPF
-             21 'CPGRRT DIREKT CHAIN, '
-             42 'UPDAT, WRITE, DELET  '
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 1                   SÄTZE
-      SATZ     80
-      ON 11 #  90 'nicht gefunden'
-      SERVICE 100
```

```
-      INFO    120 BLANK-AFTER-OUTPUT
-      X       125 EDITCODE Z

- FILE PRINTER  SPACE 2 1                  SUMME
-              17 '*** Anzahl Sätze'
-      COUNT    23 EDIT  Z

- FILE CPGRRT            ON NOT 11        RRTUPD
-      SATZ     80

- FILE CPGRRT       ADD                   RRTADD
-      SATZ     80

- FILE CPGRRT       DEL  ON NOT 11        RRTDEL
```

Beispiel B16 : Laden einer RRDS - Datei vom Kartenleser.

```
- OPTIONS  BATCH  TITEL Laden#RRDS#vom#Leser  PHASE BSPB16.

- FILES READER PRINTER CPGRRT

- INPUT DIVISION

-    FILE READER
-       1  80  SATZ

- PROCEDURE DIVISION

-    EXCPT  KOPF
-    DO LOOP
-       READ READER
-       ON EOF  BREAK
-       EXCPT SÄTZE
-    ENDDO
-    EXCPT ENDE

- OUTPUT DIVISION

- FILE PRINTER  SPACE # 2  SKIP 01        KOPF
-             23 'CPGRRT vom Leser laden'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 1                 SÄTZE
-      SATZ   80

- FILE PRINTER  SPACE 2 1                 ENDE
-             17 '*** Ende ***     '

- FILE CPGRRT   ADD                       SÄTZE
-      SATZ   80
```

Erläuterungen:

Mit oben abgebildetem Beispiel werden Karten von einem Kartenleser gelesen und in die RRDS-Datei 'CPGRRT' übertragen. Die gelesenen Sätze werden mit 'ADD' hinzugefügt, d. h. wenn bereits Sätze auf der Datei existieren, werden neue Sätze am Ende der Datei angefügt.

Beim Laden der Datei wird gleichzeitig auf der Datei 'PRINTER' ein Protokoll gedruckt.

Beispiel B17 : Sequentielles Update einer RRDS-Datei.

```
- OPTIONS  BATCH  TITEL sequentielles#UPD  PHASE BSPB17.

- FILES CPGRRT PRINTER

- -D.
-     COUNT  5 0
-     OC  2
-     RECN  3 0
- -I.
-     FILE CPGRRT
-        1 80  SATZ

- -C.
-     EXCPT KOPF
-     RECN = 1
-     DO WHILE CPGFRC = '  '
-        RECN  READ  CPGRRT
-        IF CPGFRC = '  '
-           COUNT = COUNT + 1
-           EXCPT SÄTZE
-        ENDIF
-     ENDDO
-     EXCPT  SUMME

- -O.
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGRRT'
-      UDATE  70
-      UTIME  80

- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80

- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      COUNT  23 EDIT  Z

- FILE CPGRRT                       SÄTZE
-             40 '<--- SEQ.UPD --->'
```

Erläuterungen:

In obigem Beispiel werden Sätze sequentiell in einer RRDS-Datei upgedated und gleichzeitig als Liste auf dem Drucker ausgegeben.

Die Verarbeitung beginnt mit dem 1. Satz der Datei.

Beispiel B18 : Datei mit OPEN im Programm

```
- OPTIONS  BATCH  TITEL Eintrag#No#open  PHASE BSPB18.

- FILE CPGESD  INP  VAR  8100  ESDS   NO OPEN
- FILE PRINTER

- -D
-    COUNT  5 0
-    KEY  4
-    OC  2
- -I
-    FILE CPGESD
-       1 80  SATZ

- -C
-    IF CONDITION U1.             * UPSI 1
-       OPEN CPGESD
-    ENDIF
-    IF CPGFRC =  '  '.           * File Return-Code für OPEN
-       EXCPT KOPF
-       FILL X'00' KEY
-       DO WHILE CPGFRC = '  '
-          KEY READ CPGESD
-          IF CPGFRC = '  '
-             COUNT = COUNT + 1
-             EXCPT SÄTZE
-          ENDIF
-       ENDDO
-       IF CONDITION U1.          * UPSI 1
-          CLOSE CPGESD
-       ENDIF
-       EXCPT  SUMME
-    ELSE.                        * Fehler beim OPEN
-       IF CPGFRC = 'EF'.         * Leere Datei
-          EXCPT LEERE-DATEI.     * Meldung auf den Printer
-       ELSE
-          EXCPT OPEN-ERROR.      * Meldung auf den Printer
-       END
-    ENDIF
- -O
- FILE PRINTER  SPACE # 2  SKIP 01  KOPF
-             17 'Sätze von CPGESD'
-      UDATE  70
-      UTIME  80
- FILE PRINTER  SPACE # 2  SKIP 01  LEERE-DATEI
-             22 'Datei CPGESD ist leer'
- FILE PRINTER  SPACE # 2  SKIP 01  OPEN-ERROR
-             24 'OPEN-Fehler bei CPGESD:'
-      CPGFRC 27
- FILE PRINTER  SPACE # 1           SÄTZE
-      SATZ   80
- FILE PRINTER  SPACE 2 1           SUMME
-             17 '*** Anzahl Sätze'
-      COUNT  23 EDITCODE  Z
```

Erläuterungen:

Die Datei CPGESD ist mit NO OPEN in der Files Division vereinbart. So-

mit wird die Datei nicht automatisch eröffnet, sondern kann gezielt mit dem Befehl OPEN eröffnet werden. In diesem Beispiel wird das OPEN nur ausgeführt, wenn der Schalter U1 (= UPSI 1) gesetzt ist.
OPEN setzt wie alle Dateioperationen gegebenenfalls einen Schalter und immer den File-Return-Code CPGFRC, auch dann, wenn die Datei leer ist. Bei einer leeren Datei wird zusätzlich noch der Schalter EF gesetzt.
Mit dem Befel CLOSE sollte die Datei bei Ende der Verarbeitung abgeschlossen werden.

Beispiel B19 : Ausgabe auf Stanzer

```
- OPTIONS  BATCH  TITEL Ausgabe#auf#Stanzer  PHASE BSPB19.

- FILES READER PUNCHER

- -I
-    FILE READER
-       1  80  SATZ

- -C
-    DO LOOP
-       READ READER
-       ON EOF  BREAK
-       EXCPT
-    ENDDO

- -O
-    FILE PUNCHER
-       SATZ     80
```

Beispiel B20 : Kopieren Platte Band

```
- OPTIONS  BATCH  TITEL Copy#Platte-Band  PHASE BSPB20.

- FILE IJSYS04 INP FIX   #   80  DISK
- FILE TAPE    OUT FIX 1600  80  TAPE SYS010

- INPUT DIVISION
-  FILE IJSYS04
-     1 80 SATZ

- PROCEDURE DIVISION
-  DO LOOP
-     READ IJSYS04
-     ON EOF  BREAK
-     EXCPT
-  ENDDO

- OUTPUT DIVISION
-  FILE TAPE
-     SATZ  80
```

Stichwortverzeichnis 9900

A

ACCEPT	Datensatz direkt lesen (= CHAIN)		4010
ADD	addieren (RPG-Schreibweise)		4015
addressability	Errors in der Assembly		6930
Adressierung	Adressierungs-Copy Book (Options)		3300
AFOOT	Mittelwert einer Feldgruppe rechnen		4020
Allgemeines	allgemeines		1200
Alpha	alphanumerische Felder	3952	2141
AND	Und-Verknüpfung in IF,OR und WHEN-Oper.	2453	4365
AND	Beispielprogramm		8150
Anzeige	der Data Dictionary-Texte		2620
Arithmetik	arithmetische Formeln		4850
Assembler	alternative Assemblierprozedur		3301
Assembler	Programmsequenzen in Assemblersprache		4027
Assembler	Statement ins Programm einfügen (MACRO)		4420
Assemblerliste	Assemblerliste, Aufbereitung (Options)		3300
Attribute	Tabelle		7020
Attribute	CICS-Attribute (Options)		3300
Attribute	Bildschirmattribute		3950
Aufbereitung	Aufbereitungsschlüssel (Edit-Codes)		7070
Aufbereitung	Feldaufbereitung mit EDIT	4205	8040
Ausrichtung	auf Wortgrenzen bei der Datenbeschreibung		3550
Auxiliary	Storage-Bereiche auf Platte auslagern		2190
AVERAGE	Mittelwert einer Feldgruppe rechnen		4024

B

BA	Bedingungsabfrage in der Input Division		3700
Batch	Batchprogrammierung mit HL1		6300
Batch	Batchprogramme (Options)		3300
Batch	Batch-Beispielprogramme		8600
BD	Bedingung setzen (Procedure Division)		3810
Bedingung	Bedingung abfragen (Procedure Division)		3810
Bedingung	Bedingung setzen (Procedure Division)		3810
BEGAS	Beginn eines assembler-codierten Programmteils		4027
BEGDT	Beginn Entscheidungstabelle		4030
Begriffe	Begriffsbestimmungen		7200
BEGSR	Beginn eines Unterprogramms		4035
Beispiele	Beispielprogramme		8000
Bildschirm	Ein- / Ausgabe ohne QSF		2150
Bildschirm	zwischenspeichern von Daten auf Bildschirm		2160
Bildschirm	Inhalt zurückladen mit LOADT		4405
Bildschirm	Inhalt retten mit SAVET		4640
Bildschirm	Bildschirm-Attribute		7020
Binär	binäre Speicherung		2143
Bit prüfen	mit TEST-BIT oder TESTB		4740
BITOF	Bitschalter löschen		4040
BITON	Bitschalter setzen		4045
Blockdiagramm	Blockdiagramm		2240
Blocklänge	Blocklänge in der Files Division		3400
Boolesche Verkn.	Boolesche Verknüpfung von Opeationen		2475
BREAK	Eine DO-Schleife beenden	2455	4050
Buffer Mode	drucken im Buffer Mode		8035

C

CAB	vergleichen und verzweigen		4055
CALL	Unterroutinen aufrufen (CALL-Schnittstelle)		4060

CALLD	Datasetzugriff			4063
CALLM	Unterprogramm aus der Methodenbank abrufen			4065
CAS	vergleichen und in Subroutine verzweigen			4075
CASE	vergleichen und in Subroutine verzweigen			4075
Catal	CATAL Karte über JOB CONTROL (Options)			3300
CBS	vergleichen und in Subroutine verzweigen			4080
CHAIN	Satz wahlfrei lesen			4085
CHANG	Feldinhalte austauschen			4090
CHANGE	Feldinhalte austauschen			4090
CHECK	Dateistatus prüfen			4095
Checkliste	Programmierer-Checkliste			2246
CLEAR	Feldinhalte löschen			4100
CLEAR-IND	Schalter abhängig von Index löschen			4105
CLOSE	Datei abschließen			4110
CLRIN	Schalter abhängig von Index löschen			4115
COM-REG	Communication Region ansprechen			4125
Common Area	Zwischenspeichern in der Common Area		8130	2162
COMP	vergleichen			4120
COMPARE	vergleichen			4120
COMRG	Communication Region ansprechen			4125
CONTINUE	Eine DO-Schleife unterbrechen		2457	4130
CONVERT	alphanumerischen Feldinhalt konvertieren			4135
CONVERT	Beispiele			8120
CONVT	alphanumerischen Feldinhalt konvertieren			4135
COPY	Assembler Unterprogramm einfügen			4140
CPGCOM	Zwischenspeichern in der Common Area			8130
CPGDAI	CPG-internes Datumsfeld (ISO-Datum)			2134
CPGFIS	Lesen von Dateien mit > 8K Satzlänge			8090
CPGFRC	File Return-Code			2131
CPGMPF	Programmfunktionstaste bei Operation Map			2130
CPGMRC	MAP Return-Code 'No input/Invalid Character'			2130
Cross Reference	Cross Reference listen (Options)		2247	3300
CSA	zwischenspeichern von Daten in der CSA			2160
Cursor	Beispiel variable Cursorpositionierung			8065
Cursor	Beispiel Cursor-Stop			8051
CWA	Common Work Area			7050
D				
Data Dictionary	im Programm			2600
Data Dictionary	Texte mit anlisten			2620
Data Division	Data Division			3500
Dataset	Dataset Logik im HL1 (Options)	5090	3304	3302
Datei	Beschreibung in der Files Division		3400	3450
Dateiänderung	Beispiel für Dateiänderung mit UPDAT / WRITE			8015
Dateiänderung	Beispiel für KSDS-Dateiänderung mit EXCPT			8017
Dateiänderung	Beispiel für RRDS-Dateiänderung			8020
Dateiänderung	Beispiel für ESDS-Dateiänderung		8030	8025
Dateiänderung	Beispiel Update bei variabler Satzlänge			8050
Dateianzeige	Beispiel für Dateianzeige mit READ-PAGE			8010
Dateiart	Dateiart in der Files Division			3400
Dateiname	Dateiname			2301
Dateiname	Dateiname in der Files Division			3400
Dateiorganisation	Dateiorganisation in der Files Division			3400
Dateiverarbeitung	mit CPG2			2300
Daten	Datenfelder			2100
Datenkanal	im HL1			5020
Datenstruktur	Datenstrukturen		3710	2477
Datenstruktur	Datenstruktursubfeld-Bestimmungen			2480
Datenview	Definition, Realisierung, Verarbeitung			2340
Datenview	anlegen (Vorgehensweise und Beispiel)			7530
Datum	Tagesdatum anzeigen			2134

DEBUG	Testhilfe	4150	2810
Definition	Felddefinition in der Data Division		3500
DELC	Zeichen aus Alphafeld entfernen		4155
DELET	Satz von einer Plattendatei löschen		4160
DELETE	Satz von einer Plattendatei löschen		4160
DEQ	Programmteil entriegeln		4165
DEQUEUE	Programmteil entriegeln		4165
Dezimalstellen	Dezimalstellen in der Data Division		3500
Dialog	dialogorientierte Dateipflege		8101
Dimension	einer Feldgruppe ermitteln (SETIX)		4670
direkt löschen	in einer KSDS-Datei im Batch		8608
direkt updaten	in einer KSDS-Datei im Batch		8605
Direktzugriff	auf Dateien		2306
Disk	Platteneinheit	3405	3400
Display	Bildschirmeinheit		3400
DISPLAY	Konsolmeldung ausgeben		4170
DIV	dividieren (RPG-Schreibweise)		4175
Division	Programmabschnitt		3030
Divisionsrest	übertragen mit MVR oder MOVE-REST		4505
DLI	DL/I Aufruf		4180
DL1	DL1 Datenbank		2330
DL1	variable Einträge bei DLI-Operationen		2335
DN	Dateiname		3710
DO	Befehlsfolge ausführen	2460	4185
DO UNTIL	Befehlsfolge ausführen		2461
DO WHILE	Befehlsfolge ausführen		2462
Doppelwortgrenze	Ausrichtung in der Data Division		3550
Drucken	Beispiel für Drucken im Line Mode/Buffer Mode		8035
Drucker	Druckerart in der Files Division		3400
Drucker	Druckersteuerung (Forms Division)		3600
DSPLY	Konsolmeldung ausgeben		4195
Dummywort	Dummywort (Begriffserklärung)		7200
Dummywort	Dummywort (Übersicht)		7010
Dump	Special Terminal Dump	4200	2830
DY	Dummywort (Procedure Division)		3810

E

EDIT	Alphafeld aufbereiten	8040	4205
Edit-Code	Tabelle		7070
Edit-Code	Aufbereitungsschlüssel		3900
EDN	VBOMP- und EDN-Verarbeitung	4817	4818
EF	Schalter End of File		2261
EG	Ergebnis (Procedure Division)		3810
Eingabe	Input Division		3700
Eingabeschalter	in der Input Division		3750
Einheit	Ein- Ausgabeeinheit in der Files Division	3405	3400
Einschränkungen	Einschränkungen	6310	2900
ELIM	Zeichen durch Blank ersetzen		4210
ELIMINATE	Zeichne durch Blank ersetzen		4210
ELSE	ELSE-Zweig der IF-Operation		4215
END	Ende eines strukturierten Programmblocks		4220
ENDAS	Ende eines assembler-codierten Programmteils		4027
ENDDO	Ende einer DO-Schleife		4225
ENDDT	Ende einer Entscheidungstabelle		4230
ENDIF	Ende eines IF-Blocks		4235
ENDSR	Ende eines Unterprogramms		4240
ENQ	Programmteil sperren		4245
ENQUEUE	Programmteil sperren		4245
Entscheidung	Entscheidungstabellen		2410
EOF	Schalter End of File		2261
EREAD	formatisieren und lesen des Bildschirms		4250

Ergebnis	Ergebnis (Procedure Division)		3810
ESA	Vorbereitungen auf den ESA-Mode		2970
ESDS	VSAM-ESDS-Datei		3400
ESDS	Beispiele zur Online-Verarbeitung	8030	8025
ESDS	Beispiele zur Batch-Verarbeitung	8612	8613
EVALUATE	Mehrfachauswahl	4255	2466
EXCPT	verzweigen in die Output Division		4260
EXECUTE	anderes Programm ausführen (= EXPR)		4265
EXHM	HL1-Modul ausführen		4270
EXHM-VAR	EXHM mit variablem HL1-Modulaufruf		4273
EXIT-SEND	Programm (später) an anderes Terminal schicken		4310
EXIT-TRANS	anderes Programm aufrufen mit Bildschirmdaten		4315
EXITD	anderes Programm aufrufen mit Datenübergabe		4275
EXITI	anderes Programm aufrufen		4280
EXITP	anderes Programm aufrufen		4285
EXITP-VAR	EXITP mit variablem Programmnamen		4290
EXITS	Programm (später) an anderes Terminal schicken		4295
EXITT	anderes Programm aufrufen mit Bildschirmdaten		4300
EXITT-VAR	taskorientierter Aufruf mit var. Trans-Id		4305
EXPR	anderes Programm ausführen		4320
EXPR-VAR	EXPR mit variablem Programmnamen		4325
EXSR	Subroutine ausführen		4330
externe Felder	Beschreibung in der Data Division		3540
F			
Fehler	Fehlermeldungen CPG2 während der Umwandlung		6900
Fehler	Fehlermeldungen CPG2 während der Ausführung		6950
Feld	Feldnamen (Regeln)	7015	2110
Feld	Feld (Definition)		3500
Feld	Feld (Begriffserklärung)		7200
Feldaufbereitung	mit den Befehlen EDIT und SELECT	3969	2420
Feldbestimmung	(Input Division)		3700
Felddefinition	explizit und implizit		2110
Feldgruppe	Regeln und Verarbeitung		2120
Feldgruppe	ausgeben		3965
Feldgruppe	Mittelwert berechnen mit AFOOT o. AVERAGE		4020
Feldgruppe	durchsuchen mit LOKUP oder LOOK-UP		4410
Feldgruppe	verschieben mit ROLL oder ROLLB		4625
Feldgruppe	sortieren mit SORT oder SORTA		4705
Feldgruppe	Summe errechnen mit XFOOT		4830
Feldgruppe	Begriffserklärung		7200
Feldlänge	Feldlänge in der Data Division		3500
Field	Schlüsselwort FIELD in der Input Division		3700
File	Schlüsselwort FILE in der Input Division		3700
Files Division	Files Division		3400
FILL	Alphafeld mit einem Zeichen füllen		4335
FIND	durchsuchen einer Datenview	8070	4340
fließendes	Währungszeichen		3972
Formeln	arithmetische Formeln		4850
Forms	Forms Division (Formularsteuerung)		3600
Formular	Forms Division (Formularsteuerung)		3600
Funktionstasten	Programmfunktionstasten		2260
F1	Faktor 1 (Procedure Division)		3810
F2	Faktor 2 (Procedure Division)		3810
G			
gemischte	Verwendung von festem und freiem Format		3020
GET-UPDATE	Datensatz direkt lesen (= CHAIN)		4085
GETCHANNEL	Kanal erneut übertragen		4342
GETHS	Kanal erneut übertragen		4342
GETIN	Schalter der höheren Stufe abfragen		4345

GETMI	Schalter der höchsten Stufe abfragen			4350
Gliederung	Gliederung eines Programms			3030
GO	verzweigen			4360
GO TO	verzweigen			4360
Grammatik	Grammatik der Procedure Division			3810
Grammatik	Regeln			7300
große Programme	OPTIONS-Parameter BIG, 12K und ADD		6930	3300
Gruppenstufe	Gruppenstufen			2270
H				
Halbwortgrenze	Ausrichtung in der Data Division			3550
Hauptprogramm	HL1 Hauptprogramm			3300
hexadezimal	ausgeben			3921
HL1	HL1 Dataset		5090	3400
HL1	HL1 Datenkanal			5020
HL1	HL1 Library (Options)			3300
HL1	HL1 Modul ausführen mit EXHM			4270
HL1	Programmierung			5000
I				
IF	Wenn-Abfrage	8150	2466	4365
IF CONDITION	Bezugszahlen abfragen		2467	4365
IF-DAT	Datumsfelder vergleichen, Format ttmmjj			4368
IF-DATI	Datumsfelder im ISO Format vergleichen			4370
IF-DATK	Datumsfelder mit Kalendertag vergleichen			4373
Index	abhängig von einem Schalter setzen (SETIX)			4670
indizierbar	indizierbare Operationen			2120
INDOF	Bezugszahlen löschen			4375
INDON	Bezugszahlen setzen			4380
Input Division	Input Division			3700
intern	CPG-interne Felder			2130
ISAM	ISAM - Dateien			3402
ISO-Datum	im Feld CPGDAI			2134
J				
JLB	linksbündig verschieben, Blanks nach hinten			4382
JRB	rechtsbündig verschieben, Blanks nach vorn			4385
JRC	rechtsbündig verschieben, Zeichen nach vorn			4390
JRZ	rechtsbündig verschieben, Nullen nach vorn			4395
K				
Kanal	Kanalvorschub (Forms Division)			3600
Kenntnisse	erforderliche Kenntnisse			1810
Kleinbuchstaben	Kleinbuchstaben nicht übersetzen (Options)			3300
Kommentar	Kommentare			3010
Konsolausgabe	mit DISPLAY			4170
Konstanten	maximale Länge von Konstanten			7040
Konvertieren	von Feldinhalten (Beispiele)			8120
Kopieren	KSDS nach KSDS im Batch			8604
Kopieren	Kartenleser nach ESDS im Batch			8610
KSDS	VSAM-KSDS-Datei			3400
KSDS	Datei verändern (Beispielprogramm)			8017
KSDS	Verarbeitung im Batch			8605
KW	Schlüsselwort in der Input Division			3700
L				
Laden	vom Leser in eine KSDS-Datei			8603
LEFT-SHIFT	linksbündig verschieben, Blanks nach hinten			4397
Lichtstift	Lichtstiftauswahl			2490
Lichtstift	Lichtstiftbezugszahl in der Input Division			3750
Limits	Maximalwerte eines Programms			7040

Line Mode	Beispiel für Drucken im Line Mode			8035
LIST	Ausgabe extern beschriebener Listen			4400
LIST-VAR	variabler LIST-Befehl			4401
Listbild	des Data Dictionary im Programm			2680
Liststeuerung	mit EJECT, LIST und NOLIST			2245
LOADT	geretteten Bildschirm zurückladen			4405
Logisch	logisch gepackte Speicherung			2144
Logische	Verknüpfung von IF-Abfragen		8150	4365
LOKUP	suchen in einer Feldgruppe			4410
LOOK-UP	suchen in einer Feldgruppe			4410
M				
MACRO	Assembler-Instruktion einfügen			4420
MAP	Map lesen, transaktionsorientiert			4425
MAP-VAR	MAP mit variablem Mapnamen			4426
MAPD	Map Dialog			4430
MAPD-VAR	MAPD mit variablem Mapnamen			4431
MAPI	Map einlesen			4435
MAPI-VAR	MAPI mit variablem Mapnamen			4436
MAPO	Map ausgeben			4440
MAPO-VAR	MAPO mit variablem Mapnamen			4431
MAPP	Map auf einen Drucker ausgeben			4445
MAPP-VAR	MAPP mit variablem Mapnamen			4446
Maps	QSF-Maps listen (Options)			3300
Maximalwerte	Tabelle der Maximalwerte			7040
MLLZO	ändern der Zonenhalbbytes			4450
MOVE	rechtsbündig übertragen			4455
MOVE-ARRAY	Feldgruppeninhalt übertragen			4460
MOVE-LEFT	linksbündig übertragen			4465
MOVE-REST	Rest einer Division übertragen			4505
MOVE-RIGHT	rechtsbündig übertragen			4455
MOVEA	Bereich linksbündig übertragen			4460
MOVEL	linksbündig übertragen			4465
MOVEN	Alphafeld in numerisches Feld übertragen			4470
MOVER	rechtsbündig übertragen			4455
MOVEV	variables MOVE			4475
MULT	multiplizieren (RPG-Schreibweise)			4500
MVR	Divisionsrest übertragen			4505
N				
Namen	Feldnamen			2110
Negativbeträge	in Schablonen			3959
Nullen	Nullenunterdrückung per Schablone			3957
Numerisch	numerische Felder		3954	2142
O				
Oberfläche	Benutzeroberfläche (siehe Surface)			3010
OC	Operations Schlüsselwort (Procedure Division)			3810
OE	erweiterter Operationscode (Procedure Division)			3810
OF	Schalter Overflow			2261
OK	Operatoren (Procedure Division)			3810
ON	Bedingung abfragen (Procedure Division)			3810
OP	Operations Schlüssel (Procedure Division)			3810
OPEN	Datei eröffnen			4515
OPEN	per Programm im Batch (Beispiel)			8619
Operation	Übersicht der Operationen			7000
Operation	Operation (Procedure Division)			3800
Operationscode	erweiterter Operationscode			3810
Operationscode	Übersicht		7410	3800
Operatoren	Operatoren (Procedure Division)			3810
Optimierung	Optimierungsfunktion des CPG	2500	2670	3510

Optimierung	für numerische Operationen (Options)		3304
Options Division	Options Division		3300
OR	Oder-Verknüpfung in IF,DO und WHEN-Oper.	2468	4365
OR	Beispielprogramm		8150
ORG	Positionieren in der TWA		3520
Output Division	Output Division		3900
Overflow	Overflow-Abfrage	8600	2261
Overlay	Feldüberlagerung	2304	2485
P			
PARAMETER	Parameter übergeben (CALL-Schnittstelle)		4517
PARM	Parameter übergeben (CALL-Schnittstelle)		4517
PERFORM	Unterprogramm ausführen (= EXSR)		4520
PF-Tasten	Programmfunktionstasten		2260
Platten-Änderung	Beispiel für Änderungen in Platten-Dateien		8000
Printer	Drucker Datei		3400
PRNT	Assembler Listausgabe		4525
Procedure	Procedure Division		3800
Programm	Programmfunktionstasten		2260
Programmgröße	OPTIONS-Parameter BIG, ADD und 12K	6930	3300
PROT	Schutzcode vergeben (für CPG3-Anwender)		4527
PROTECTION	Schutzcode vergeben (für CPG3-Anwender)		4527
PURGE	Temporary Storage Queue löschen		4530
PUTIN	Schalter zu höherer Stufe übertragen		4535
PUTMI	Schalter zur höchsten Stufe übertragen		4540
Q			
QLF	Quick List Facility		2550
QSF-Maps	QSF-Maps listen (Options)		3300
QSSA	Qualifiziertes SSA aufbauen (DL/I)		4545
Quadratwurzel	ziehen mit SQARE-ROOT oder SQRT		4710
Queuing	Temp Storage Queuing in der Files Division		3400
Queuing	Beispiel für Temporary Storage Queuing	2195	8060
R			
Random	automatisches Random all (Options)		3300
RANDOM	Datei freigeben		4620
READ	lesen		4555
READ-BACK	Datei rückwärts lesen		4565
READ-BACK	Beispiel für eine Dateianzeige mit READ-BACK		8045
READ-PAGE	von einer Datei in eine Seite einlesen		4580
READ-PAGE	Beispiel für Dateianzeige mit READ-PAGE	8010	8040
READB	Datei rückwärts lesen	8609	4565
READB-PAGE	Datei rückwärts in eine Seite einlesen		4570
Reader	Leser in der Files Division		3400
READI	Bildschirm transaktionsorientiert lesen		4575
READP	von einer Datei in eine Seite einlesen		4580
RECEIVE	taskorientiertes Lesen einer Map		4425
Redefinieren	von Feldern in der Data Division	3510	3520
Referenzstruktur	im Programm		2640
Regeln	Regeln für Feldnamen		2110
Regeln	Regeln für Feldgruppennamen		2120
Releaseänderung	CPG2		0040
REPLACE	Zeichen durch anderes Zeichen ersetzen		4600
REPLC	Zeichen durch anderes Zeichen ersetzen		4601
Reserviert	reservierte Feldnamen		7015
RIGHT	rechtsbündig verschieben, Blanks nach vorn		4385
RIGHT-CHAR	rechtsbündig verschieben, Zeichen nach vorn		4390
RIGHT-ZERO	rechtsbündig verschieben, Nullen nach vorn		4395
RNDOM	Datei freigeben		4620
ROLL	Feldgruppe verschieben		4625

ROLL-BACK Feldgruppe rückwärts verschieben 4630
ROLLB Feldgruppe rückwärts verschieben 4630
RRDS VSAM RRDS-Datei 3400
RRDS Beispiel zur Verarbeitung einer RRDS-Datei 8614 8020

S
Satz Satz (Syntax) 3010
Satz Satzbeschreibung (Input Division) 3700
Satz Satzbestimmung (Input Division) 3700
Satz Satz (Begriffserklärung) 7200
Satzende Satzende-Zeichen 3010
Satzformat Satzformat in der Files Division 3400
Satzlänge Satzlänge in der Files Division 3400
SAVET Bildschirminhalt retten 4640
SCAN Alphafeld nach einer Zeichenkette durchsuchen 4645
Schablonen Schablonen 3956
Schablonen Beispiele für Schablonen 3963
Schalter (Bezugszahlen) 3810 2260
Schalter abhängig von einem Index löschen (CLRIN) 4115
Schalter abfragen mit IF CONDITION 4365
Schalter abfragen auf höherer HL1-Stufe (GETIN) 4345
Schalter abfragen auf höchster HL1- Stufe (GETMI) 4350
Schalter löschen (Gruppe oder alle) mit INDOF 4375
Schalter setzen (Gruppe oder alle) mit INDON 4380
Schalter übertragen zur höheren HL1-Stufe (PUTIN) 4535
Schalter übertragen zur höchsten HL1-Stufe (PUTMI) 4540
Schalter abhängig von einem Index setzen (SETIN) 4665
Schalter löschen (bis zu drei) mit SETOF 4680
Schalter setzen (bis zu drei) mit SET oder SETON 4685
Schleife Schleife beenden mit BREAK 4050
Schleife Schleife programmieren mit DO-Befehlen 4185
Schleife Schleife unterbrechen mit CONTINUE 4130
Schlüssel bei Dateiverarbeitung 2304
Schlüssellänge Schlüssellänge in der Files Division 3400
Schlüsselwort Schlüsselworte der Service-Funktion 3810
Schlüsselwort Schlüsselworte der Procedure Division 7410
Schlüsselwort Schlüsselwort (Begriffserklärung) 7200
Schutzstern Schutzsternschreibung 3970
SCREENDUMP = SDUMP 4650
SDUMP Special Terminal Dump (Testhilfe) 4650 2830
Segment in Verbindung mit CHAIN (U) 8145
Segment in Verbindung mit READI 8150 4575 8140
SELCT Feldauswahl (Umkehrung der Feldaufbereitung) 4655
SELECT Feldauswahl (Umkehrung der Feldaufbereitung) 4655
Semicolon Semicolon als Satzende-Zeichen (Options) 3300
SEND Ausgabe einer Map (= MAPO) 4440
sequentiell sequentieller Dateizugriff 2306
sequentiell löschen in einer KSDS-Datei im Batch 8607
sequentiell updaten in einer KSDS-Datei im Batch 8606
Service Service Funktionen 2000
Service Service-Schlüsselworte 3810
SET Schalter setzen 4685
SET-INDEX Index abhängig von einem Schalter setzen 4670
SET-INDIC Schalter abhängig von einem Index setzen 4665
SET-LIMIT Zeiger auf einen Satz einer Datei setzen 4675
SETIN Schalter abhängig von einem Index setzen 4665
SETIX Index abhängig von einem Schalter setzen 4670
SETLL Zeiger auf einen Satz einer Datei setzen 4675
SETOF Schalter löschen 4680
SETON Schalter setzen 4685
SORT Feldgruppe sortieren 4705

SORTA	Feldgruppe sortieren		4705
Speicherung	Speicherungsarten von Datenfeldern		2140
Sperren	eines Programmteils mit ENQ / DEQ		4245
Sprache	Sprache der Diagnostik (Options)		3300
SQARE-ROOT	Quadratwurzel ziehen		4710
SQL	Beispiel		7570
SQRT	Quadratwurzel ziehen		4710
Standardvorz.	Standardvorzeichen (Options)		3300
starre Notation	gemischt mit freier Schreibweise		3020
Statement	Anweisung (Syntax)		3010
Statement	Anweisung (Begriffserklärung)		7200
Storage	Temporary Storage in der Files Division		3400
Strukturierte	strukturierte Programmierung		2450
SUB	subtrahieren (RPG-Schreibweise)		4715
Subroutine	ausführen mit EXSR oder PERFORM		4330
Surface	Benutzeroberfläche (Syntax)		3010
Surface	Benutzeroberfläche (Options)		3300
SV	Serviceabfrage (Procedure Division)		3810
SYNCP	Synchronization Point spezifizieren		4720
SYNCPOINT	Synchronization Point spezifizieren		4720
Syntax	die Syntax der Sprache CPG2		3010
Syntax	Syntax Fehler		6900
System	Systeminformationen mit COMRG CPGSIN		4125
T			
Tabellen	Tabellen		7000
TAG	Merkmal setzen		4730
taskorientiert	transaktionsorientierte Dateipflege		8110
TCT	Terminal Control Table		7055
TCT	zwischenspeichern von Daten in der TCT		2160
TDUMP	= SDUMP		4650
Telefon	Telefon-Service		0030
Temporary Storage	allgemein		2190
Temporary Storage	in der Output Division		3900
Temporary Storage	Queue löschen mit PURGE		4530
Temporary Storage	Tabelle		7060
Temporary Storage	Beispiel für Temporary Storage Queuing	2195	8060
TEST-BIT	Bit prüfen		4740
TEST-Field	Feld auf numerische Zeichen prüfen		4762
TEST-NUM	Feld auf numerische Zeichen prüfen		4745
TEST-TERM	Bildschirmnamen prüfen		4750
TESTB	Bit prüfen		4740
TESTF	Feld auf numerische Zeichen prüfen		4743
Testhilfen	DEBUG und SDUMP		2800
TESTN	Feld auf numerische Zeichen prüfen		4745
TESTT	Bildschirmnamen prüfen		4750
Texte	aus dem Data Dictionary anlisten		2620
TIME	Zeit setzen		4780
Titel	Programm-Titel (Options)		3300
transaktions	orientierte Dateipflege (Beispielprogramm)		8110
transaktions	orientierte Programmierung		2550
Transient Data	zwischenspeichern von Daten auf TD		2160
Transient Data	Transient Data in der Files Division		3400
TWA	Transaction Work Area (Options)		3300
TWA	TWA-Überlagerung	2485	3510
TWA	TWA-Überlagerung mit ORG		3520
TWA Größe	OPTIONS-Parameter ADD, 12K und BIG	6930	3300
TWA-LOAD	TWA von Temporary Storage einlesen	4785	8080
TWA-SAVE	TWA auf Temporary Storage retten	4790	8080

TWALD	TWA von Temporary Storage einlesen	4785	8080
TWASV	TWA auf Temporary Storage retten	4790	8080
U			
UCTRN	übersetzen in Großbuchstaben		4805
UDATEC	CPG-internes Feld mit aktueller Jahreszahl		2134
UDATEI	CPG-internes Datumsfeld		2134
Überlagerung	Überlagerung in der TWA	3510	2485
Überlagerung	durch Positionieren mit ORG		3520
überlappt	überlappte Felder auf dem Bildschirm		3967
Überlauf	Steuerung (Beispiel)		8600
übersetzen	in Großbuchstaben mit UCTRN		4805
Uhrzeit	Uhrzeit anzeigen		2134
Umwandlung	CPG-Umwandlung ohne IJSYS04		7540
unformatiert	unformatierte Ausgabe		3900
ungepackt	ungepackte numerische Werte		2145
UPDAT	Datensatz auf Datei/Storage zurückschreiben		4810
UPDATE	Datensatz auf Datei/Storage zurückschreiben		4810
Update	Beispiel für Update bei variabler Satzlänge		8050
UPSI	UPSI-Schalter bei Batch-Programmierung	6335	2260
USSA	unqualifiziertes SSA aufbauen (DL/I)		4815
V			
Variabel	var. Einträge bei DLI-Operationen		2335
Variabel	var. Dateiname für Drucker, TS und TD		3400
Variabel	variabler Mapname	8075	4425
variabel	variable Satzlänge, Verarbeitungsbeispiel		8050
varibable EXHM	EXHM mit variablem HL1-Modulaufruf		4273
VBOMP	VBOMP- oder EDN-Verarbeitung	4817	4818
Verriegeln	eines Programmteils mit ENQ / DEQ		4245
View	Definition, Realisierung, Verarbeitung		2340
Vollwortgrenze	Ausrichtung in der Data Division		3550
Voraussetzungen	Voraussetzungen für die Programmierung		1810
Voraussetzungen	Software-Voraussetzungen		1820
VSAM	VSAM-Dateien		2315
VSLCT	VBOMP-Daten in die TWA übertragen		4818
W			
Währungszeichen	fließendes Währungszeichen		3972
wahlfrei	wahlfreier Zugriff auf Dateien		2306
WAIT	warten		4820
Warning	Warnung nach der CPG-Umwandlung		6920
WCC	Write Control Character Tabelle		7030
WHEN	WHEN-Bedingung der EVALUATE-Operation	4822	2470
WHEN OTHER	WHEN OTHER-Bedingung der EVALUATE-Operat.	4823	2471
Working Storage	Section (= Data Division)		3500
Wort	CPG Wort (Syntax)		3010
Wort	CPG Wort (Begriffserklärung)		7200
WRITE	Datensatz auf Datei/Storage neu anlegen		4825
X			
XFOOT	Summe einer Feldgruppe rechnen		4830
Z			
Z-ADD	löschen und addieren (RPG-Schreibweise)		4835
Z-SUB	löschen und subtrahieren (RPG-Schreibweise)		4840
Zonenhalbbyte	bei (ungepackten) numerischen Werten		2145
Zonenhalbbyte	ändern mit MLLZO		4450
Zubehör	Hilfsprogramme		7500

Zuweisen von Werten 4850
Zwischenspeichern von Daten 2160