QPG

Programmierer-Handbuch

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

Lattwein-Informationen

Dieses Handbuch wird als Anleitung für die Verwendung des QPG ... Quick Program Facility herausgegeben.

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

QPG-Änderungen Release 2.6

Allgemein

... Standard ist jetzt extended PWA mit 32767 Byte, kann aber mit "N" auf 8192 Bytes wie im CPG bzw. wie im QPG bis zur Version 2.5 reduziert werden, und zwar im QTF bei der Einrichtung der Library (siehe QTF-Handbuch, Seite 3111).

... Bei Syntaxfehlern werden jetzt EH-Wert und Farbe aus den QTF-Standardwerten genommen.

... Die Performance wurde weiter optimiert bei Eingabe/Ausgabe und Datenkanal.

... Die Universial Database UDB wurde im VSE unterstützt.

Options

... Bei OPTIONS IO ist die Ausgabe mit den Operationen UPDATe/WRITE in der Input und Output Division identisch. Betrifft numerische ungepackte Felder.

Procedure Division

... CALL und CPARM zum Aufruf von externen Unterprogrammen.

... DELC kann mehrere Zeichen aus einem Feld entfernen.

... ELIM kann mehrere Zeichen in einem Feld löschen.

... ENQ/DEQ mit Namen sind unterstützt.

... MOVEV mit Service N funktioniert wie MOVEL, wenn das Ergebnisfeld alpha ist.

... PURGE bei DATASETS mit CPGFRC = 'P' unterstützt.

... REPLC kann mehrere Zeichen durch ein anderes ersetzen.

Output Division

... Bei variabel langen TS-Queues wird die Satzlänge geprüft.

... Bei Ausgabe variabel langer TS-Queue-Sätze und CPGVRL=0 wird als Satzlänge die maximale IO-Position verwendet.

Debug Programm

... Bei DEBUG wird das aktuelle Programm mit Taste F12 beendet.

Servicetransaktion QPG

... Newcopy funktioniert auch, wenn Library einen Usecount > 1 (maximal 5) hat.

Batch Utility

... Im QPGUTIL kann eine Vorlaufkarte simuliert werden. Übergabe von bis zu 61 Bytes Daten in PRDATA, siehe S.6201.

Fehler

... Bessere Diagnostik bei ABENDs – der Programmname wird mit ausgegeben.

... Alle gemeldeten Fehler im QPG wurden behoben.

Kurzbeschreibung

QPG ist ein Just-in-Time-Compiler und eine Weiterentwicklung des CPG.

Die Sprache ist weitestgehend mit dem CPG2-Format kompatibel, verzichtet allerdings auf veraltete Funktionen wie Schalter, festes Format, Bit-Operationen und Assembler Statements. Dadurch werden die Programme verständlicher und leichter wartbar.

Verschiedene funktionale Verbesserungen und Erweiterungen werden nur im QPG und nicht mehr im CPG angeboten. Dadurch kommt es tendenziell zu Inkompatibilitäten: QPG-Programme können nicht grundsätzlich mit den Compilern CPG und HL1 umgewandelt werden.

QPG ist objektorientiert und kann sowohl im Umfeld der klassischen CPG-Anwendungen (online und im Batch) als auch in Browseranwendungen unter CICS oder im Zusammenspiel mit Sprachen anderer Plattformen und der Software CPGXML eingesetzt werden. Grundsätzlich erbt QPG die Eigenschaften des rufenden Objekts, übernimmt automatisch Daten vom rufenden Objekt und übergibt nach der Verarbeitung wiederum automatisch Daten an das rufende Objekt.

Allgemeines

Es können standardmäßig bis zu je 1.000 Test- und Produktionslibraries eingerichtet werden. Die Libraries werden mit Ausnahme der standardmäßig vorhandenen Libraries PROG, QPG, TASK und TEST automatisch beim ersten Aufruf in der Directory verzeichnet. Eigene QPG-Libraries können hierdurch einfach angelegt werden. Die Pflege der Directory QPGPXD entfällt.

Sprachenabhängige Syntax

Die Syntax ist auf die Umgebung in der QPG eingesetzt wird angepasst. In der englischen Version gelten für QPG folgende Vereinbarungen:

Zeilenende	;	(deutsche Version .)
Dezimalzeichen	.	(deutsche Version ,)
Kolonnentrennzeichen	,	(deutsche Version .)

Die englische Version wird in der Kundenkonfiguration (CPGURSIT, Stelle 10) mit dem Eintrag '.' als Dezimalzeichen gesetzt.

Für QPG-Programme kann im QTF beim Anlegen einer Library oder eines Dokumentes die Syntax angepasst werden. Hiermit können die Standards für das Statementende, das Dezimalzeichen und das Trennzeichen bei numerischen Konstanten (Tausenderpunkt) geändert werden. Eine Angabe ist nur erforderlich, wenn die aktuell angezeigten Standards modifiziert werden sollen. Werden diese Standards in der Library geändert, dann gelten diese für alle Dokumente der Library. Diese Standards können auch noch für einzelne Dokumente geändert werden. Diese haben dann Vorrang vor den Library-Standards bzw. den Einstellungen bei der CPG-Installation.

Erweiterte Adressierung

Für neue QPG-Programme oder Libraries kann automatisch die erweiterte Adressierung aktiviert werden. Damit kann jedes QPG-Programm bis zu 32 kByte an privaten Daten verarbeiten. Diese Adressierung ist erforderlich, um alle neuen Funktionen nutzen zu können. Bei bestehenden QPG-Anwendungen kann die erweiterte Adressierung auf Library- oder Dokumentenebene aktiviert werden.

Tipp: Rufen Sie im QTF mit Auswahl 'B' die Bibliothekenverwaltung auf und setzen für Ihre QPG-Libraries den Parameter Erweiterte PWA auf 'X', damit in allen Programmen die erweiterte PWA genutzt werden kann. Falls gewünscht, kann in einzelnen Programmen im QTF mit Auswahl 'A' die Erweiterte PWA mit 'N' abgeschaltet werden.

Programmierung

QPG-Programme werden im Textsystem QTF als Programm-Dokumente gespeichert. Aus QTF können QPG-Programme direkt umgewandelt und getestet werden. Im QTF können verschiedene Programm-Libraries benutzt werden, wie z. B. PROG und TEST.

In jeder Programm-Library können Programme als QTF-Dokumente angelegt werden. Die einzelnen Programme werden in freier Schreibweise erfasst. Felder, Operationen, Dateien usw. können groß oder klein eingegeben werden. Sie werden bei der Umwandlung übersetzt. Bei alphanumerischen Konstanten erfolgt keine Übersetzung, sie werden so eingegeben, wie sie z. B. am Bildschirm oder auf der Liste erscheinen sollen. Bei Returncodes z.B. ist daher auf die richtige Angabe der Konstanten ('EF' statt 'ef' usw.) genau zu achten.

Beim Umwandeln jedes Programms erfolgt eine Diagnostik, bei der Syntaxfehler direkt am Bildschirm angezeigt werden und dort auch korrigiert werden können.

Bei der Umwandlung wird ein Pseudo-Code generiert, der von QPG interpretiert und ausgeführt wird. Der Pseudocode ist so konzipiert, dass das Programm mit maximaler Performance ausgeführt wird und annähernd die Geschwindigkeit von kompilierten CPG-Programmen erreicht. Gleichzeitig ist der Pseudocode so kompakt, dass für jedes Programm nur ein minimaler Speicherbedarf erforderlich ist. Selbverständlich sind QPG-Programme reentrant und werden nur einmal in der CICS-Umgebung gespeichert.

Syntax

Der allgemeine Aufbau eines Programms besteht aus:

OPTIONS	oder
FILEs	-F
DATA DIVISION	-D
INPUT DIVISION	-I
PROCEDURE DIVISION	-C
OUTPUT DIVISION	-O

Diese Reihenfolge der Statements muss eingehalten werden. Es ist mindestens 1 Statement in der Procedure Division erforderlich. OPTIONS, DATA, INPUT und OUTPUT Division sind wahlweise.

Es ist nur das CPG2-Format (freie Schreibweise) unterstützt. Im QTF-Programmdokument werden die Spalten 1-72 benutzt. Es wird normalerweise kein '-' in Spalte 6 verwendet. Alternativ kann aber auch das CPG2-Format benutzt werden, bei dem alle Statements ein '-' in Spalte 6 enthalten. In diesem Format erfolgt die Codierung in den Spalten 8-79.

Die Spalte 72 muss leer sein. Ausnahme sind SQL-Statements bei denen die Spalte 72 bei Fortsetzungszeilen mit '+' markiert wird. Hier muss die Spalte 71 leer bleiben.

Es ist nur ein Statement pro Zeile erlaubt.

Kommentarzeilen werden durch einen '*' gekennzeichnet. Ein Statement wird durch einen '.' (deutsche Version) oder ';' (englische Version) beendet. Danach kann ein Kommentar folgen, der mit einem '*' beginnen sollte. Kommentare vor der Spalte 72 sollten mit '*' beginnen.

Je nach Operation sind Dummyworte zur Dokumentation erlaubt, z.B.: BY FROM INTO IS THAN THEN usw. Dummyworte sind nicht als Operanden zugelassen.

Großschreibung ist nicht erforderlich, mit Ausnahme von Konstanten, wenn es dort gewünscht oder notwendig ist, wie z.B. beim File Return Code CPGFRC.

Ein Programm kann beliebig in Seiten aufgeteilt werden. Hierdurch lassen sich Programme besser dokumentieren.

Beim ersten Aufruf eines Programms wird die Syntax geprüft. Es erfolgt ein Abbruch, sobald ein Fehler festgestellt wird. Der Fehler wird im Klartext zusammen mit dem fehlerhaften Statement angezeigt. Das Programm kann danach mit QTF korrigiert werden.

Wird ein Fehler nicht korrigiert und das Programm dennoch ausgeführt, so erfolgt ein Abbruch mit dem Code 'ECPR'.

Options

Es sind folgende Optionen unterstützt:

CLEAR Die Option CLEAR ermöglicht die Abfrage der CLEAR-Taste nach den Operationen EXHM und TASK. Ohne diese Option wird durch die CLEAR-Taste die aktuelle Transaktion im CICS beendet.

DAT Dataset-Logik. Die Felder des Programms bleiben innerhalb der Transaktion bestehen. DAT oder PWA ist erforderlich, wenn Ein- und Ausgabeoperationen benutzt werden.

Die Geschwindigkeit bei der Ausführung ist optimiert. Beim ersten Aufruf eines QPG-Programms wird ein Fast Data Channel aufgebaut. Dadurch werden bei Rücksprung und jedem Folgeaufruf direkt die Felder übertragen, ohne die Namentabellen abzugleichen. Hierdurch reduziert sich besonders bei Batchanwendungen die Laufzeit wesentlich.

DEBug Debugging. Hiermit wird die Online-Testhilfe aktiviert.

DEF Define Felder. Alle Felder werden generiert, selbst wenn diese nicht benutzt werden. Das automatische Unterdrücken der nicht benutzten Felder wird hiermit aufgehoben.

DIC Mit der Option DICtionary wird bei langen Feldnamen geprüft, ob das Feld in den Data Dictionary Standards angelegt ist. Ist das Feld hier nicht definiert, dann erzeugt QPG bei der Umwandlung eine Fehlermeldung, während sonst der Feldname nach 6 Stellen abgeschnitten wird.

Mit OTIONS DICtionary werden auch Feldnamen mit weniger als 7 Stellen aus dem zentralen Data Dictionary in den internen Kurznamen übersetzt, wenn diese Sonderzeichen enthalten.

END Ende der Optionen. Mit einem '.' oder dem Parameter END werden die Optionen beendet. Werden nach '.' oder END weitere Optionen gefunden, so erfolgt eine Fehlermeldung.

NODEBug No Debug wird angegeben, um für das Programm die Debug-Funktion auszuschalten. Dies gilt auch dann, wenn DEBUG von außen mit der Transaktion QPG gesetzt wird, oder wenn im Programm die Operation DEBUG verwendet wird.

HTML Hyper Text Markup Language. Mit dieser Option kann das QPG-Programm aus einer Internet-/Intranet-Anwendung aufgerufen werden. Hierbei werden automatisch die Daten mit dem Inter-/Intranetserver ausgetauscht.

Die internen Felder CPGDAT, CPGPRL, CPGTIM, UDATE, UDAY, UMONTH, UTIME und UYEAR sind hierbei implizit definiert, d.h. diese Felder brauchen nicht durch OPTIONS DEFine definiert werden, um sie in NetPage-Masken benutzen zu können. Dies gilt allerdings nur, wenn die Übertragung nicht durch das interne Feld CPGEDS begrenzt ist.

Bei OPTIONS HTML wird sowohl die komprimierte, als auch die einfache QPCF-Storage-Queue für die Anzeige im Browser oder an CPGXML-Anwendungen übertragen.

Bei OPTIONS HTML können die internen Felder UDATE und UTIME auch dann in einer NetPage-Maske angezeigt werden, wenn im Programm der Datenkanal mit CPGEDS begrenzt wurde.

IO **Die Ausgabe mit den Operationen UPDATe und WRITE ist damit in der Input- und in der Output-Division identisch. Ansonsten gibt es Unterschiede bei ungepackten numerischen Feldern. Falls mit Input- und Output-Bestimmungen für ein und dieselbe Datei gearbeitet wird, ist dieser OPTIONS-Parameter zu empfehlen.**

PFK Program Function Keys. Dieser OPTIONS-Parameter wird angegeben, um das interne Feld CPGMPF auch nach einer EXHM-Operation zu aktualisieren. Hiermit kann die Taste abgefragt werden, die zuletzt im Modul (Dialog) betätigt wurde. Der OPTIONS-Parameter PFK entspricht der Verwendung der PF-Tasten in einem CPG-Programm und dient der Kompatibilität.

PWA Save Private Work Area. Innerhalb der Task bleibt die PWA erhalten, Felder werden jedoch, anders als bei DAT, bei jedem Aufruf neu initialisiert.

Wie bei DAT ist die Geschwindigkeit bei der Ausführung optimiert.

Die OPTIONS-Parameter können auf mehrere Zeilen verteilt werden. Mit einem '*' können Kommentare benutzt und von den Options abgetrennt werden.

Weitere Parameter können benutzt werden, um z. B. ein fertiges Programm mit dem Compiler CPG2 oder HL1 umzuwandeln. Hierbei erfolgt vom QPG keine Syntaxprüfung.

Beispiele:

```
OPTIONS  DATASET
         DEBUG
         DEFINE
         END.
```

```
OPTIONS  HTML              * Kommentar Programm für Inter/Intranet
         PWA.
```

```
OPTIONS  DAT DEF END PFK   * Dieses Statement ist falsch, da END nicht letzter
                           * Parameter ist.
```

Files

Alle im QPG benutzten Dateien müssen im Data Dictionary File beschrieben sein.

Es ist bei FILE nur der Dateiname erforderlich. Zusätzlich kann z.B. für Batchanwendungen als Service INP, OUT oder UPD angegeben werden.

Außerdem kann der Parameter TYPE xx angegeben werden, wenn die Datei unter einer bestimmten Satzart im DD beschrieben ist.

Bei Dateien wird die im Data Dictionary definierte Einheit geprüft. Bei Einheiten, die nicht unterstützt sind erfolgt eine Fehlermeldung. Die Einheit 'DISK' ist nur mit der Dateiart K=KSDS, E=ESDS, R=RRDS und der Dateiorganisation V=VSAM zugelassen. Z.Zt. sind folgende Dateitypen unterstützt:

HL1DS	HL1-Datasets
VSAM	KSDS-, ESDS- und RRDS-Files mit fester oder variabler Satzlänge
PROG	QPG-Datasets
STORAGE	Temporary Storage mit fester oder variabler Satzlänge
TABLE	Find-Tabellen

Die FILE-Section kann (muss aber nicht) mit -F beginnen.

Beispiele:

```
-F
        FILE ARTIKEL
        FILE AUFTRAG INP
        FILE KUNDEN  DD TYPE DS
        FILE POSTEN  DD TYPE 01 INP
```

Für Dateizugriffe sind die OPTIONS DAT oder PWA erforderlich.

Folgende Dateioperationen sind gültig:

Opcode/Art	VSAM	HL1DS	PROG	STORAGE	TABLE
CHAIN	X	X	X		
CHECK	X	X	X		
CLOSE	X	X	X		
DELET	X	X	X		
FIND					X
OPEN	X	X	X		
PURGE				X	
READ	X	X	X	X	
READB	X	X	X		
READI	X	X	X	X	
RNDOM	X	X	X		X
SETLL	X	X	X	X	
UPDAT	X	X	X	X	
WRITE	X	X	X	X	

Einträge im Data Dictionary

Einheit / Library	HL1DS	KSDS	ESDS	RRDS	PROG (Libr)	STORAGE	TABLE
Dateiname	x	x	x	x	x	x	x
Ein-/Ausgabe I,U,O	x	x	x	x	x	x	I
Satzformat F,V	F	F,V	F,V	F	F	F	F
Satzlänge	x	x	x	x	x	x	x
Dateiart		K	E	R	(S)		
Dateiorganisation		V	V	V	P	(A)	
Schlüssellänge		x	4				
Schlüsselposition		x					
TS-Verarbeitung						Q,S	
var. Verarbeitung						(V)	

erforderliche Einträge x

optionale Einträge:

(Libr) = QPG-Library
(S) = Special Dataset (Input/Output Statements werden benutzt)
(A) = Auxiliary Storage (Daten werden auf Platte ausgelagert)
(V) = Variabler Name (wird in CPGTSN bereitgestellt)

Special QPG-Datasets

Wird im Data Dictionary in Dateiart ein 'S' eingetragen (Special), dann erfolgt die Datenübertragung in den Ein- und Ausgabebestimmungen. Damit ist die Verarbeitung genau so wie bei VSAM-Dateien. Dadurch lassen sich Datasets erstellen, die sowohl in QPG-, HL1- als auch in RPG-Anwendungen aufgerufen werden können.

Dieser Typ ist konzipiert für die einfache Migration von VSAM-Dateien in QPG-Datasets.

Bei Special Datasets kann ein Key in Faktor 1, z.B. bei CHAIN angegeben werden. Der Inhalt wird dabei im internen Feld CPGKEY übergeben.

Key

Auf KSDS-Dateien kann auch mit einem gepackten Key zugegriffen werden, wenn der Schlüssel in gepacktem Format gespeichert ist. Es ist zu beachten, dass das Key-Feld bei positiven Werten nach einer Rechenoperation die Zone 'C' und nach einer MOVE-Operation die Zone 'F' hat. Bei KSDS-Dateien wird die Schlüssellänge bei der Umwandlung geprüft.

Auf ESDS-Dateien wird mit der relativen Byte-Addresse und auf RRDS-Dateien mit der relativen Satznummer zugegriffen.

Bei Datasets wird in der (z.B. READ- oder CHAIN-)Operation kein Keyfeld angegeben, da der Key automatisch (bei HL1 über den Datenkanal und bei QPG über den Feldnamen) übergeben wird. Ausnahme: Special Datasets (Eintrag 'S' im Data Dictionary bei Dateiorganisation). Hier wird der Key linksbündig im internen Feld CPGKEY übertragen.

Temporary Storage Queues werden mit der Satznummer verarbeitet. Ein Update auf einen bestimmten Satz einer TS-Queue kann erfolgen, indem mit der Satznummer positioniert wird (SETLL) und anschließend das Update erfolgt. Bei einer READ- statt einer SETLL-Operation würden die Felder durch die Eingabe wieder überschrieben werden.

Bei Temporary Storage sind nur Queues und simulierte Queues (TS-Verarbeitungsart S) unterstützt.

Ist bei TS im DD Organisation 'A' angegeben, dann wird eine TS-Queue bei WRITE und UPDATE auf Auxiliary Storage ausgegeben.

Variabler TS-Name

Ist ein TS-Bereich im DD mit 'V' (var. Name) gekennzeichnet, dann kann der Name im internen Feld CPGTSN (8 Stellen alpha) vorgegeben werden. Ist das Feld CPGTSN im Programm nicht angegeben, dann wird der Eintrag 'V' für die variable Verarbeitung ignoriert. Das Programm arbeitet dann so wie in QPG-Version 2.0. Ist CPGTSN angegeben und wird jedoch nicht gefüllt, dann wird der Dateiname als TS-Name benutzt. Achtung: Hierbei ist der Storage allerdings nicht terminal-abhängig.

Variable Satzlänge

VSAM-Dateien und Temporary Storages können mit variabler Satzlänge verarbeitet werden. Hierzu muss im Data Dictionary bei Dateibeschreibung das Satzformat 'V' eingetragen werden. Nach Lesebefehlen wird die effektive Satzlänge (ohne Praefix) im Feld CPGVRL übergeben, wenn es definiert wurde. Bei der Ausgabe kann CPGVRL verwendet werden, um eine Satzlänge vom Programm vorzugeben. Hierzu muss in der Output-Division zusätzlich der Parameter 'VAR' angegeben werden.

Bei Dateiausgabe mit dem Parameter VAR wird geprüft, ob das Feld CPGVRL definiert ist. Wenn nicht wird eine Fehlermeldung ausgegeben. Die variable Satzlänge muss einen gültigen Wert enthalten.

Strings

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 freige-geben wird.

Data Division

In der Data Division werden die Datenfelder eines Programms definiert, sofern es keine internen Felder sind, die automatisch verfügbar sind, oder sofern die Felder nicht in der Input Division definiert sind.

Beispiele:

```
    DATA DIVISION
         FELD               20.         * alpha
         INDEX               3 0.       * num.
         WERT               11 2.       * num.
         FG           10 *  79.         * Feldgr. alpha
         FN           10 *   6 0.       * Feldgr. num
         SATZ          0 *  80.         * wird überlagert
         SFG          80 *   1.         * von Feldgruppe
         DEFINE KUNDEN.                 * Data Dictionary
    ORG  FELD.                          * Redefine Feld
         HW                  2 HALF.    * Halfword
         FW                  4 FULL.    * Fullword
         DW                  8 DOUBLE.  * Double word
    ORG.                                * Reset Location
         DEFINE AUFTRAG TYPE 01.        * DD Satzart
         DEFINE KDNADR MULtiple.        * nur neue Felder anlegen
         CPGEDS        0 *   1.         * Beginn der 'privaten' Felder
         I                   3 0.       * Index
         WF                 20.         * Workfield
         DATE                8.         * ttmmjjjj
    ORG  DATE CHK.                      * Redefine Feld mit Check
         TT                  2.         * tt
         MM                  2.         * mm
         JJ                  2.         * jj
         XYZ                 3.         * ===> Syntaxfehler bei CHK
```

Schlüsselworte:

CHK	bei ORG prüft, ob eine Überlagerung passt.
DEFINE	Felder aus Data-Dictionary-Strukturen definieren.
DOUble	Ausrichten auf Doppelwortgrenze.
FULl	Ausrichten auf Vollwortgrenze.
HALf	Ausrichten auf Halbwortgrenze.
ORG	Redefinition eines Feldes oder einer Feldgruppe.
MULtiple	Definieren von Feldern, die noch nicht angelegt sind.
TYPE	Angabe der Satzart im Data Dictionary bei DEFINE.

Work Area

Die Felder des Programms werden in einer privaten Work Area gespeichert. Diese Work Area (PWA) wird wie bei HL1-Modulen dynamisch alloziert und kann maximal 32 kBytes an Userdaten speichern. Durch Überlagerung mit ORG kann der Platz mehrfach genutzt werden.

CPGEDS

Dieses interne Feld steuert den Datenaustausch.

DEFINE

Bei DEFINE werden Überlagerungen berücksichtigt. Dies gilt auch dann, wenn die Struktur nicht lückenlos im Data Dictionary beschrieben ist. Dies gilt allerdings nur für Alphafelder und numerisch gepackte Felder.

Es werden auch DEFINE-Strukturen richtig verarbeitet, die unvollständig erfasst wurden. Eine ungewollte Überlagerung mit anderen Feldern wird dadurch vermieden.

Es wird schon bei Definition eines Feldes erkannt, wenn es vorher definiert war. In der Diagnostik wird dann das betreffende Statement angezeigt.

Wird bei der Definition MULtiple verwendet, dann werden nur Felder angelegt, die noch nicht im Programm definiert sind. Die Meldung ‚Feld doppelt definiert' wird damit unterdrückt.

Mit DEFINE kann auch eine Struktur definiert werden, die aus nur einem Feld besteht.

ORG

Ein Feld kann mit ORG redefiniert werden. Das nächste Feld beginnt dann mit der Startposition des redefinierten Feldes. Wird CHK angegeben, dann muss das letzte Feld der Überlagerung auf genau der gleichen Position enden wie das überlagerte Feld – ansonsten bricht der Compiler die Umwandlung mit einer Fehlermeldung ab. Mit einer weiteren ORG-Anweisung innerhalb der Überlagerung wird die Prüfung mit CHK wieder ausgeschaltet.

Compiler-Prefix

Ist im DD bei der Datei ein Compiler-Praefix angegeben, dann wird bei den Feldern dieser Praefix vorangestellt. Der Feldname ist dabei aber auf 6 Stellen begrenzt. Das gilt natürlich auch für DD-Definitionen in der Input Division.

Input Division

In der Eingabe werden Dateien, Strukturen (Satzarten), HL1-Datenkanäle und Feldselektionen beschrieben.

Beispiele:

```
    DATA DIVISION
          CPGIOA  1200 *  1.                    * IO-Area für QPG-Dataset
          PRDATA  8 * 1
    INPUT DIVISION
      FILE  KUNDEN DD.                          * aus Data Dictionary
      FILE  AUFTRAG.
                    1    2   SA
         SEGMENT AUFKOPF DD TYPE 01 REF AUFTRAG. * aus DD, Satzart 01
         SEGMENT AUFPOS  DD TYPE 02 REF AUFTRAG.
         SEGM    AUFTEXT DD TYPE 03 REF AUFTRAG.
      FILE  HHELP  HS.                          * HL1-Storage
                    1    8   DOKUM
      FIELD CPGCOM.                             * Common Area
                   21   28   NKEY
      FIELD CPGCOM TYPE T1 DEF
                    1    1   A
      FIELD CPGCOM SELECT-TYPE T2
                    2    2   B
      FIELD CPGIOA
                 1200 1200   F1200
      FIELD CPGSIN.                             * Systeminformationen
                    1    8   APPLID
      FIELD AREA
                    1    2 0 N2
              BIN   3    4 0 N4
        SEGMENT ARTIKEL
              PAC   5    6 1 N31
        SEGM KUNDE
              LOG   7   10 2 N82
      ARRAY PRDATA
        1  8  PARAM.                            * vergleiche Seite 6201
```

Hinweis:

Die Datei AUFTRAG hat 3 Satzarten (AUFKOPF, AUFPOS, AUFTEXT). Beim sequentiellen Zugriff muss zunächst die Satzart analysiert werden, um bei einem Wechsel der Satzart die Verarbeitung beenden zu können. Deshalb ist zunächst nur das Feld SA (Satzart) beschrieben. Je nach Inhalt des Feldes SA erfolgt dann im Programm die Eingabe der entsprechenden Satzart mit der Operation READI.

Ein SEGMENT-Name kann bis zu 8 Stellen lang sein. Eine Segment-Definition kann auch mit dem Schlüsselwort SEGM (statt SEGMENT) beginnen.

Interne Felder wie z.B. UDATE oder CPGTID dürfen nicht in der Input-Division eingelesen werden.

Ein Strukturname darf nicht mehrfach vorkommen. Z.B. darf ein Feld- oder Segmentname nicht identisch sein mit einem Dateinamen. Ausnahme ist, wenn DD angegeben und in der vorherigen Struktur der gleiche Name benutzt wurde.

Beispiel:

```
    PROCEDURE DIVISION
         DO LOOP
             READ AUFTRAG
             IF CPGFRC = 'EF'
                BREAK
             END
             IF SA = '01'
                READI AUFTRAG AUFKOPF
             END
             IF SA = '02'
                READI AUFTRAG SEGM AUFPOS
             END
             IF SA = '03'
                READI AUFTRAG SEGMENT AUFTEXT
             END
             *             ... Verarbeitung
         END

         SELECT CPGCOM TYPE T1
         SELECT CPGCOM SELECT-TYPE T2
```

Bei der Eingabe wird die maximale Satz- bzw. Feldlänge geprüft. Bei Segment-Definitionen muss die zugehörige File-Definition vorher codiert sein, damit die richtige Länge geprüft wird. Bei HL1-Datenkanälen ist die maximale Position 3944.

Strukturen

Zu einer Feld-, Datei- oder Datenkanal-Definition können mehrere Strukturen aus dem Data Dictionary zusammengefasst werden. Folgestrukturen beginnen dabei mit dem Schlüsselwort 'DD'. Beispiel:

```
-I
         FILE AUFTRAG DD TYPE 01
                      DD TYPE 02
                      DD REF AUFTEXT
                      DD REF AUFBEST TYPE 03
```

Define für WRITE und UPDATE

QPG enthält standardmäßig eine Optimierungsfunktion, bei der alle Felder, die nur in der Input Division definiert sind und die im Programm nicht verwendet werden, entfernt werden. Hiermit erfolgt eine Optimierung hinsichtlich Speicherbedarf und Performance. Die Optimierung wird ausgeschaltet bei MAP- und LIST-Operationen, oder wenn DEFine in der OPTIONS-Anweisung angegeben ist.

Die Optimierung kann auch gezielt für einzelne DD-Strukturen ausgeschaltet werden, wenn hierbei nach DD der Parameter DEF oder DEFINE angegeben wird. Dies ist bei den Operationen WRITE und UPDATE erforderlich, wenn die Ausgabe über die Input Division erfolgt und die Datenfelder nicht explizit definiert sind.

Beispiele:

```
INPUT DIVISION
         FILE STOR01 DD DEFINE
         FILE DATEI2 DD DEF TYPE 01
         SEGM STRUC1 DD TYPE 01 DEFINE
         FILE CPGWKV DD TYPE 04 REF QTFTXT DEF
         FILE DATEI3 DD REF QTFTXT DEF TYPE 03
         FIELD XXXXX DD REF CPGWRK TYPE 07 DEF
                     DD REF CPGWRK TYPE 08 DEF
                     DD REF CPGWRK DEF
PROCEDURE DIVISION
         WRITE STOR01
         UPDATE DATEI2
```

Der Parameter DEF oder DEFINE kann an beliebiger Stelle nach DD angegeben werden. Er darf in einem Statement jedoch nicht mehrfach benutzt werden.

DEF oder DEFINE kann ebenfalls angegeben werden, um Felder in Strukturen zu definieren, die explizit (mit Von- und Bis-Name) beschrieben sind.

Beispiele:

```
INPUT DIVISION
         FILE STOR DEFINE
                     1  100   SATZ
         SEGM STRUC2 DEF
                    11   15   PLZ
         FIELD SQLCAF DEF
                   116  120   SQSTAT
```

Hinweis: Bei HL1-Datenkanälen erfolgt keine Optimierung. Alle Felder, die in ei-
einem Kanal (HS) definiert wurden, sind damit automatisch definiert.
Der DEF- oder DEFINE-Parameter ist hierbei nicht erforderlich.

Variable Satzlänge

Bei VSAM-Dateien, die mit variabler Satzlänge definiert sind, wird nach einer Eingabeoperation (READ, READB oder CHAIN) die Satzlänge im Feld CPGVRL übergeben. Das gleiche gilt ebenfalls für variabel lange Temporay Storages. Wird ein Feld länger eingelesen als die Satzlänge erlaubt, dann hat der Rest des Feldes einen undefinierten Inhalt und muss gegebenenfalls im Programm gelöscht werden.

CPGSIN

Mit SELECT CPGSIN werden erweiterte Systeminformationen abgerufen:

```
INPUT DIVISION

      FIELD CPGSIN
                   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 Security 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 Area.
              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.     * aktuelle Anzahl Bildschirmzeilen
              PAC 62  64 0 ACTCOL.     * aktuelle 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 Bildschirmeingabe
                  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   DATUM1.     * Datum JJ.TTT
                 109 116   DATUM2.     * Datum JJ.TT.MM
                 117 124   DATUM3.     * Datum JJ.MM.TT
                 125 132   DATUM4.     * Datum TT.MM.JJ
                 133 140   DATUM5.     * 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

PROCEDURE DIVISION

         SELECT CPGSIN
```

Procedure Division

Folgende Operationen sind unterstützt:

=	*	Zuweisung
+	*	Addieren
*	*	Multiplizieren
-	*	Subtrahieren
/	*	Dividieren
ACCEPT		sh. CHAIN
AFOOT		Arithmetischer Mittelwert
AVERAGE		sh. AFOOT
BEGSR		Beginn Unterprogramm
BREAK		Abbruch einer DO Schleife
CALL		**reserviert für internen Gebrauch** *
CHAIN		Direkter Zugriff
CHANGe	x	Tauschen von Feldinhalten
CHECK		Prüfen Dateistatus
CHECK-VAR	x	Prüfen Dateistatus variabel
CLEAR		Löschen Inhalte aller Datenfelder
CLOSE		Schließen Datei
COM-REG		sh. COMRG
COMRG		Holen Systeminformationen
COMPUTE	c	Berechnung von Formeln
CONT		Nächster Schleifendurchlauf
CONTINUE		sh. CONT
CONVERT	*	sh. CONVT
CONVT	*	Konvertieren von Feldinhalten
CPARM		**reserviert für internen Gebrauch** *
DEBUG		interaktive Testhilfe
DELC	*	Löschen von Zeichen
DELETe		Löschen von Sätzen
DEQ		Programmteil entriegeln
DISPLAY		sh. DSPLY
DO	x	Schleife
DO UNTIL	x	Schleife ausführen bis
DO UNTIL-DATe	x	Schleife mit Datumsabfrage im Standardformat TTMMJJ
DO UNTIL-DATI	x	Schleife mit Datumsabfrage im ISO-format JJMMTT
DO WHILE	x	Schleife
DO WHILE-DATe	x	Schleife mit Datumsabfrage im Standardformat TTMMJJ
DO WHILE-DATI	x	Schleife mit Datumsabfrage im ISO-format JJMMTT
DUMP		Speicherauszug und Steuern Handling bei Programmabbruch
DSPLY		für Ausgabe auf Konsole
EDIT	x	Feld aufbereiten
ELIM	*	Eliminieren Zeichen
ELIMINATE	*	sh. ELIM
ELSE		Abfrage Nein Zweig
END		Ende bei DO oder IF
ENDDO		Ende bei DO
END-EVALUATE		sh ENDEV
ENDEV		Ende einer EVALUATE Gruppe
ENDIF		Ende bei IF
ENDPR		beendet ein Programm
ENDSR		Ende Unterprogramm
ENQ		Programmteil sperren
EVALUATE		Beginn einer mehrfachen Fallunterscheidung
EXHM		Execute HL1-Modul
EXIT-TRANS		sh. EXITT

Operationen

EXITD		Transaktions-Start mit Datenübergabe
EXITI		Starten Trans-Id mit Intervall Control
EXITT		Starten einer neuen Transaktion
EXPR		**reserviert für internen Gebrauch** *
EXSR		Unterprogramm ausführen
FILL	*	Feld füllen
FIND		Durchsuchen von Tabellen
GETHS		Get Higher Storage für Datasetverarbeitung
HTMLI		Erneutes Übertragen der Daten im Intra-/Internet
HTMLO		Ausgabe einer NetPage Maske im Intra-/Internet
IF	x	Abfrage Ja Zweig
IF-DATe	x	Datumsabfrage Standardformat TTMMJJ Ja Zweig
IF-DATI	x	Datumsabfrage ISO-format TTMMJJ Ja Zweig
JLB	*	Justify Left with Blank
JRB	*	Justify Right with Blank
JRC	*	Justify Right with Character
JRZ	*	Justify Right with Zerös
LEFT-SHIFT	*	Justify Left with Blank, sh. JLB
LIST		Druckausgabe
LIST-VAR		variable List Operation
LOADT		Bildschirminhalt zurückladen
LOADT-VAR		Bildschirminhalt variabel zurückladen
LOKUP	x	Durchsuchen einer Feldgruppe
MAP		Bildschirm Datentransfer
MAP-VAR		variable MAP Operation
MAPD		Bildschirm Dialog
MAPD-VAR		variabler MAP Dialog
MAPI		Bildschirm Input
MAPI-VAR		variabler MAP Input
MAPO		Bildschirm Output
MAPO-VAR		variabler MAP Output
MAPP		Bildschirm Print
MAPP-VAR		variables MAP Printing
MOVE	*	Übertragen rechtsbündig
MOVE-ARRAY	*	sh. MOVEA
MOVE-LEFT	*	sh. MOVEL
MOVE-REST	x	sh. MVR
MOVE-RIGHT	*	sh. MOVE
MOVEA	*	Übertragen Array
MOVEL	*	Übertragen linksbündig
MOVEN		Übertragen numerisch
MOVEV		Variable Move Operation
MVR	x	Divisionsrest übertragen
OPEN		Datei öffnen
PERFORM		Unterprogramm ausführen, sh. EXSR
PROG		Programm ausführen
PROG-VAR		Variabler QPG-Programmaufruf
PROGRAM		sh. PROG
PROT		Protection Check nach neuer Logik
PROTECTION		als Synonym für PROT
PURGE		Löschen von TS-Queues
RANDOM		sh. RNDOM
READ		Datei sequentiell lesen
READ-BACK		als Synonym für READB
READB		Rückwärts lesen
READI		Satzart einlesen
RECEIVE		sh. MAP
REPLACE	*	sh. REPLC
REPLC	*	Ersetzen von Zeichen

Operationen

RIGHT	*	sh. JRB
RIGHT-CHAR	*	sh. JRC
RIGHT-ZERO	*	sh. JRZ
RNDOM		Datei freigeben
ROLL	x	Scrolling von Arrays
ROLL-BACK	x	sh. ROLLB
ROLLB	x	Scroll-Back von Arrays
SAVET		Bildschirminhalt retten
SAVET-VAR		Bildschirminhalt variabel retten
SCAN		Durchsuchen von Feldern
SCREENDUMP		sh. SDUMP
SDUMP		Anzeige Special Dump am Bildschirm
SELCT	x	Feld selektieren
SELECT	x	sh. SELCT
SEND		sh. MAPO
SET-LIMIT		sh. SETLL
SETLL		Datei positionieren
SORTa		Sortieren von Arrays
SQL		Verarbeiten einer SQL-Datenbank
SQRT	*	Square Root
SQUARE-ROOT	*	sh. SQRT
TASK		Taskorientierter Programm-Start
TASK-VAR		Variable TASK Operation
TESTF	*	Prüfen auf numerischen Inhalt
TEST-FIELD		sh. TESTF
TIME	x	Aktualisieren der Zeit
TWALD		Laden der mit TWASV gespeicherte privaten TWA
TWASV		Speichern der privaten TWA des QPG-Programms
TWA-LOAD		sh. TWALD
TWA-SAVE		sh. TWASV
UCTRAN		sh. UCTRN
UCTRN		Ein/Ausschalten der Übersetzung
UPDATe		Ändern von Sätzen
WAIT	x	Warten Zeitintervall
WHEN	x	Fallabfrage in einer EVALUATE Gruppe
WHEN-DATe	x	Fallabfrage Datum im Standardformat TTMMJJ
WHEN-DATI	x	Fallabfrage Datum im ISO-format TTMMJJ
WRITE		Anlegen von Sätzen
XFOOT		Summe einer Feldgruppe

Hinweis:

*	Full Array Support:	Hier können die Operanden auch als Feldgruppen mit oder ohne Index angegeben werden.
x	Indizierbar:	Hier können die Operanden auch als Feldgruppen mit Index angegeben werden.
c	complete arrays:	Berechnung bei COMPUTE kann auch für Feldgruppen ohne Index ausgeführt werden.

Output Division

In der Ausgabe können Felder oder Feldgruppen (alpha) aufbereitet werden, oder es werden Dateien beschrieben, die mit UPDAT geändert oder in die mit WRITE hinzugefügt werden soll. Dies gilt nicht für HL1- oder QPG-Datasets. Beispiele:

```
PROCEDURE DIVISION
         :
         EDIT FELD
         EDIT PAGE(X)
         EDIT INFO TYPE ERR01
         EDIT ZEILE SEGMENT KOPF
         EDIT ZEILE SEGM POSTEN
         UPDAT ARTIKEL
         UPDAT KUNDEN SEGM ADRE
         WRITE AUFTAG SEGMENT POS
         WRITE STOR
         EDIT CPGIOA
         :

OUTPUT   DIVISION
         FIELD FELD
                             1 '*'.            * Konstante
                   X         5.                * ungepackt
                   Y         6 BIN.            * binär
         FIELD PAGE
                   I         9 LOG BLA.        * log. gepackt, Löschen n Ausg.
                   NFG(2)   12 PAC.            * gepackt
                   J        20 EDIT Z BLA.     * Edit-Code Z, Löschen n. Ausg.
                   FELD     40 BLANK.          * Alphafeld, Löschen nach Ausg.
                   WERT     60 EDI M '*'.      * mit Schutzstern
                   WERT2    70 EDI M '$'.      * mit fl. Währungszeichen
                   DATUM    80 EDI Y '*'.      * Datum mit Nullenunterdrückung
         FIELD INFO TYPE ERR01
                            15 'Falsche Taste: '
                   CPGMPF   17
         FIELD INFO TYPE ERR02
                            16 'Kunden-Nr. fehlt'
         FIELD ZEILE SEGMENT KOPF
         FIELD ZEILE SEGM POSTEN
         :
         FIELD CPGCOM
                   FGN    4080 EDI J.          * num. Feldgruppe
         FILE  CPGWRK
                            100 '*'.            * Konstante
         FILE  ARTIKEL
                   MENGE   132 PAC.            * gepackt
         FILE  STOR
                   SATZ     80.                * Alphafeld
         FILE  CPGWKV VAR
                   DATEN   280.                * SL gesteuert über CPGVRL
         FILE  CPGKSD DD.
         FILE  STO2 DD VAR.
         FILE  KUNDEN DD
         FILE  KUNDEN DD SEGM ADRE
         FILE  AUFTRAG DD
         FILE  AUFTRAG DD SEGMENT POS REF AUFPOST TYPE 01
         FIELD CPGIOA
                            1200 '*'.           * Konstante
```

Ein Strukturname darf nicht mehrfach vorkommen. Z.B. darf ein Feld- oder Segmentname nicht identisch sein mit einem Dateinamen.

Es sind keine Schablonen unterstützt. Bei numerischen Feldgruppen beträgt die maximale Ausgabelänge 4095 Bytes.

Edit-Codes

Numerische Felder oder Feldgruppen werden mit Edit-Codes aufbereitet, z.B.:

```
-o
        field page
                  pos       edit 1
                  x         edit z
                  datum     edit y
```

Es gibt folgende Edit-Codes:

Edit-Code	Vorzeichen CR	Vorzeichen -	keins	Kolonnentrennz. Ja	Kolonnentrennz. Nein	Kolonnentrennz. Datum	führ. Nullen Ja	führ. Nullen Nein	Beispiel
A	x			x			x		1.234,56CR
B	x			x				x	1.234,56CR
C	x				x		x		1234,56CR
D	x				x			x	1234,56CR
J		x		x			x		1.234,56-
K		x		x				x	1.234,56-
L		x			x		x		1234,56-
M		x			x			x	1234,56-
X			x		x			x	12345O
Y			x			x	x		0.12.3456
Z			x		x			x	123456
1			x	x			x		1.234,56
2			x	x				x	1.234,56
3			x		x		x		1234,56
4			x		x			x	1234,56

Aufbereitung

Bei den Edit-Codes sind auch Schutzstern, fließendes Währungszeichen und Minus vor dem Feld unterstützt. Diese werden als Attribute nach dem Edit-Code angefügt.

Bei der Ausgabe können auch numerische Feldgruppen ungepackt oder mit Editcode ausgegeben werden.

Attribute

werden hinter dem Edit-Code angegeben, um Schutzsterne, das Währungszeichen oder Minus vor dem Feld auszugeben. Beispiele:

```
-o
        field page
                  betrag    edit m '$'
                  summe     edit j '*'
                  datum     edit y '*'
                  wert      edit j '-'
```

```
'*'     Schutzsterne bzw. Nullenunterdrückung bei Datum (Edit-Code Y)
'$'     Fließendes Währungszeichen:
'-'     Minuszeichen vor dem Feld bei Edit-Codes J, K, L und M:
```

Beispiele:

Wert	(Länge)	Edit-Code	Ausgabe
1.234,56-	(8,2)	J mit '*'	'**1.234,56-'
1.234,56-	(8,2)	J mit '$'	' $1.234,56-'
1.234,56-	(8,2)	J mit '-'	' -1.234,56'
0	(8,0)	Y mit '*'	' '

Datum

Je nach Kundenkonfiguration (CPGURSIT, Edit-Codes) wird das Datum auch mit '/' anstelle des Punktes aufbereitet, z.B. '24/08/2000' statt '24.08.2000'. Das Datum kann hier auf TTMMJJ oder MMTTJJ eingestellt sein.

Aufbereitung von Feldgruppen

Bei der Ausgabe einer Feldgruppe werden alle Elemente hintereinander mit je 2 Byte Abstand aufbereitet.

CPGIOA

Bei Special QPG-Datasets kann die IO-Area mit EDIT auch dann aufbereitet werden, wenn sie als Array mit mehr als 256 Byte Länge definiert wurde:

```
-D
        CPGIOA       1200 *   1.              * IO Area for Special Datasets
-C
        EDIT CPGIOA
-O
        FIELD CPGIOA
                     1200 '*'
```

Variable Satzlänge

Bei der Ausgabe (WRITE) von VSAM-Dateien mit variabler Satzlänge wird der Satz so lang angelegt, wie die höchste Ausgabeposition im Satz definiert ist. Bei temporary Storage ist in einer CICS-Umgebung ebenfalls die variable Satzlänge unterstützt. Bei Batchverarbeitung gilt die erste Satzlänge bei Ausgabe einer TS-Queue auch für alle nachfolgenden Sätze.

Wird allerdings in der Output Division zur FILE-Anweisung der Parameter 'VAR' angegeben, dann wird als Satzlänge der Wert benutzt, der im Feld CPGVRL zur Verfügung gestellt wurde (Voraussetzung: CPGVRL ist definiert und gefüllt), siehe auch Beispiel auf Seite 8070. Der Parameter VAR muss als letzter Parameter angegeben werden.

Löschen nach Ausgabe

Wird bei einem Feld, einer Feldgruppe oder einem Feldgruppenelement in der Ausgabe der Parameter 'BLAnk' angegeben, so wird es nach Ausgabe automatisch gelöscht.

Automatische Ausgabeposition

Wird in der Ausgabe die Position nicht codiert, dann wird diese von QPG automatisch errechnet. Das Feld oder die Konstante wird dann automatisch ohne Zwischenraum hinter der höchsten Ausgabeposition angehängt. Hiermit kann z.B. das Erstellen von Tables in einer Intra-/Internet-Anwendung wesentlich vereinfacht werden. Ein Mischen von Ausgaben mit fester und mit automatischer Position ist möglich.

Beispiel:

```
-o
        file qpcf segm header
                        'TABLE j;hj;agzslj;'
                        'agzhlj;agwblj;saldo;'
                        'agmslj;'
                        'emdlj;'
                        'ezglj;ewblj;esaldo;'
        file qpcf segm detail
                  ajahr
                                    ';1.HJ;'
                  avzi1     edit j
                                    ';'.
                  azah1     edit j
                                    ';'.
                  awer1     edit j
                                    ';'.
                  asal1     edit j
                                    ';'.
                  msljt1
                                    ';'.
                  aevzi1    edit j
                                    ';'.

                  aezah1    edit j
                                    ';'.
                  aewer1    edit j
                                    ';'.
                  aesal1    edit j
                                    ';'.
```

Verketten von Feldern

Ausgabebestimmungen können mit + in einer Ausgabezeile verkettet werden:

```
-o
        field date
              tt + '.' + mm + '.' + jj
```

Ausgabe-Separator

In der Ausgabe kann ein Separator definiert werden, der automatisch zwischen den einzelnen Zeilen eingefügt wird. Der Separator wird weder vor der ersten noch nach der letzten Ausgabezeile eingefügt. Der Separator wird auch nicht zwischen Elementen eingefügt, die mit + in einer Ausgabezeile verkettet sind. Durch das Verketten mit + können Trennzeichen sowohl vor der ersten, als auch nach der letzten Ausgabezeile ausgegeben werden. Durch den Separator kann z.B. die Ausgabe einer Tabelle für CPG5 vereinfacht werden. Der Separator kann als beliebig langes Alphafeld oder als maximal 6 Stellen lange Alphakonstante definiert werden. Der Separator ist nicht in Verbindung mit Data Dictionary (DD) erlaubt. Der Separator wird nach dem Schlüsselwort SEP definiert. Wird kein Feld oder keine Konstant als Separator benutzt, so wird x’00’ als Trennzeichen benutzt.

Beispiele:

```
-o
        field a sep
                  '1'
                  '2'
                  '3'
        field a type a2 sep '!'
                  'a'
                  'b'
                  'c'
        file qpcfxl sep x'09'
                  '1'
                  '2'
                  '3' + x'0d0a'
        file qpcfxl type vsep sep trz
                  trza + 'A'
                  'B'
                  'C' + trze
        file tspr sep '.'
                  uday
                  umonth
                  uyear
```

Syntaxprüfungen

Es erfolgt eine Fehlermeldung, wenn als Feldname ein reserviertes Wort benutzt wird.

In der Data Division wird die maximale Feldlänge geprüft. Diese ist bei Alphafeldern 256 Bytes und bei numerischen Feldern 15 Ziffern.

Es wird geprüft, ob die Programmabschnitte in der richtigen Reihenfolge vorliegen, d.h Options vor Files, Files vor Data Division usw.

Bei Eingabe und Ausgabe erfolgt eine Plausibilitätsprüfung. Es werden Fehler erkannt, wenn in der Eingabe ein Alphafeld gepackt eingelesen wird, oder wenn ein Alphafeld oder eine Konstante mit Edit-Code ausgegeben wird.

Bei der Eingabe wird die maximale Satz- bzw. Feldlänge geprüft. Bei Segment-Definitionen muss die zugehörige File-Definition vorher codiert sein, damit die richtige Länge geprüft wird. Bei HL1-Datenkanälen ist die maximale Position 3944.

In der Procedure Division erfolgt eine Fehlermeldung, wenn ein internes Feld wie z.B. UDATE als Ergebnisfeld benutzt wird, z.B. bei FILL ' ' TO UDATE.

In der Procedure Division werden die Services auf Gültigkeit geprüft. So ist z.B. keine MAPD- oder MAPI-Operation mit den Services C,A,K oder T erlaubt.

In der Output Division wird jetzt bei den Edit-Codes geprüft, ob die Ausgabeposition groß genug ist, um die aufbereiteten Felder aufzunehmen.

Preprocessor

Zur Erleichterung der Codierung ist im QPG ein Preprocessor (eine Art Macro-Sprache) verfügbar. Der Programmierer gibt dabei im Programmcode Anweisungen für den Preprocessor mit. Der Preprocessor erzeugt hieraus Programmcode.

Die Anweisungen für den Preprocessor beginnen mit einem '§' in Spalte 1 der betreffenden Programmzeile. Dahinter folgt der Name der Funktion, die der Preprocessor jeweils ausführen soll. Folgende Funktionen sind unterstützt:

§QSAT	QSAT-Access (für SQL-Programmierung) einfügen
§SQL	SQL-Definitionen (für SQL-Programmierung) einfügen
§TEXT	Schnittstellen zum Texsystem QTF einfügen

Hinter dem Funktionsnamen können Parameter angegeben werden. Fehlen diese Angaben, dann wird der Benutzer beim Preprocessing am Bildschirm aufgefordert, die Parameter einzugeben. Die Eingabe kann mit F3 abgebrochen werden.

Es können beliebig viele Preprocessor-Anweisungen in einem Programm enthalten sein. Das Preprocessing kann beliebig of wiederholt werden. Der Preprocessor wird aus dem QTF-Menü mit Auswahl 'X' und den Angaben bei Execute QPG : PREP aufgerufen, wobei das zu bearbeitende Programm unter Dokumentname und Library angegeben ist.

Nach erfolgtem Preprocessing kann das eigentliche Programm (gegebenenfalls mit New-Copy) ausgeführt werden.

SQL/QSAT-Preprocessor

Ein Preprocessor in der Servicelibrary QPG erlaubt es, die mit QSAT definierten ACCESSE in ein QPG-Programm einzubinden. In dem Programm wird an den betreffenden Stellen in Spalte 1 beginnend die Anweisung §QSAT definiert um dem Preprocessor mitzuteilen, dass hier ein QSAT ACCESS benötigt wird. Beispiel:

```
-D
         SQCODE              7 0.      * Define SQL Code

§QSAT    DEFINE xxxxxxxx.              * Define QSAT Access xxxxxxxx

-C

§QSAT    SELECT xxxxxxxx.              * Select für Access xxxxxxxx

         IF CPGFRC = 'SC'.             * SQL Check
            .
            .                          * Hier SQL-Fehlerbehandlung
            .
         END

         DO LOOP

§QSAT    FETCH xxxxxxxx.               * Fetch für Access xxxxxxxx

            IF CPGFRC = 'EF'.          * End of File
               BREAK
            END
            .
            .                          * Hier Daten verarbeiten
            .
         END
```

Der Preprocessor wird aktiviert mit Auswahl 'X' mit Execute QPG : PREP aus dem QTF-Menü. Die §QSAT Befehle werden vom QPG-Preprocessor aufgelöst und durch die entsprechenden expandierten SQL-Statements ersetzt.

Folgende §QSAT-Anweisungen werden unterstützt:

```
ACCESS   generiert ein SQL ACCESS Statement.
DEFINE   generiert die SQL Hostvariablen in der Data Division.
FETCH    generiert ein SQL FETCH  Statement.
INSERT   generiert ein SQL INSERT Statement.
SELECT   generiert ein SQL SELECT Statement, WHERE-Bedingung ist zu ergänzen.
UPDATE   generiert ein SQL UPDATE Statement, WHERE-Bedingung ist zu ergänzen.
```

Ist QSAT nicht installiert, so können die betreffenden Statements mit Command §SQL anstelle von §QSAT generiert werden. Werden bei SQL-Tabellen jedoch Feldnamen verwendet, die länger als 6 Stellen sind, so müssen diese im Data Dictionary definiert sein.

Wird §QSAT oder §SQL ohne Anweisung und Parameter angegeben, so wird der Bediener am Bildschirm geführt.

TEXT-Preprocessor

Der TEXT-Preprocessor in der Servicelibrary QPG erlaubt es, QTF-Funktionen auf einfache Weise im QPG-Programm einzubinden. Im Programm wird an den betreffenden Stellen in Spalte 1 beginnend die Anweisung §TEXT definiert, um dem Preprocessor mitzuteilen, dass hier eine TEXT-Funktion benötigt wird. Beispiel:

```
-d
§text    define
-c
§text    create testdok # beispiel
§text    write anfang
         fill ' ' to txrec
         do 10.
§text       write
         end
§text    write ende
§text    close testdok
§text    edit  testdok
```

Folgende §TEXT-Anweisungen werden unterstützt:

CLOSE	schließt das mit den Befehlen OPEN oder CREATE eröffnete Dokument.
COPY	kopiert ein Dokument. Parameter Dokument, Library, Dokument 2, Lib 2.
CREATE	erstellt ein Dokument. Parameter sind Dokument, Library, Description.
DEFINE	definiert Textfelder in der Data Division. ALL generiert alle Felder.
DELETE	löscht ein Dokument. Parameter Dokument, Library, (Seite, bis Seite)
DISPLAY	verzweigt ins Textsystem zur Anzeige. Parameter Dokument und Library.
EDIT	verzweigt ins Textsystem zum Ändern. Parameter Dokument und Library.
HELP	zeigt einen Hilfetext an. Parameter sind Dokumentname und Library.
OPEN	öffnet ein Dokument. Parameter Dokument, Library, (INP zur Anzeige).
PRINT	druckt ein Dokument. Parameter sind Dokument, Library und Printer-Id.
READ	liest einen Satz des mit OPEN eröffneten Dokumentes sequentiell.
UPDATE	verändert den aktuellen Satz in dem mit OPEN eröffneten Dokument.
WRITE	schreibt je einen Satz sequentiell in das mit OPEN eröffnete Dokument.

Wird §TEXT ohne Anweisung und Parameter angegeben, so wird der Bediener am Bildschirm geführt.

Parameter, die nicht gefüllt sind, können mit dem Auslassungszeichen '#' übersprungen werden.

Operationen

Bei den Operationen werden folgende Begriffe verwendet:

DY Dummyworte zur besseren Lesbarkeit des Programms, die Worte: ALL BY FROM INTO IS OFF ON THAN THEN TIMES TO und WITH sind reserviert.

EG Ergebnis als Feldname oder Feldgruppe mit oder ohne Index, je nach Operation. *

FN Filename in Dateioperationen. Die Datei muss als File definiert sein.

F1 Faktor 1 als Feld, Feldgruppe mit oder ohne Index oder als Konstante je nach Operation.

F2 Faktor 2 als Feld, Feldgruppe mit oder ohne Index oder als Konstante je nach Operation.

OC Abfrage für Vergleiche bei DO WHILE, DO UNTIL und IF Operationen. Mögliche Einträge: > < = >= => <= =< >< oder <>.

OP Operation (sh. Seite 1600).

SV Service als Ergänzung zur Operation, z.B. H oder ROU zum Runden.

Einträge im Klammern sind wahlweise und werden nur bei Bedarf angegeben, z.B:

EG = F1 * F2 (SV) hier ist der Service (ROUnded) optional.

Felder

zur Angabe variabler Daten, z.B.: KDNR, KEY, BETRAG, WERT. Ein Feldname kann maximal 30 Stellen lang sein. Ist der Feldname in den Data-Dictionary-Standards definiert und ist bei der QPG-Library auch die DD-Library (z.B. '*') angegeben, dann wird hieraus der entsprechende, maximal 6 Stellen lange Kurzname übernommen. Sonst wird der Feldname auf 6 Stellen abgeschnitten. Der Feldname muss mit einem Buchstaben A-Z oder dem $-Zeichen beginnen. In den folgenden Stellen sind auch Ziffern und _ zugelassen. Mathematische Zeichen wie z.B. +, -, *, / oder _ sind in Feldnamen nicht zugelassen.

Feldgruppen

Feldgruppen sind in einem Teil der Operationen unterstützt, sh. Seite 1600. Es kann entweder nur der Name der Feldgruppe angegeben werden, wenn die gesamte Feldgruppe (Element für Element) verarbeitet wird (Full Array Support), oder es kann ein fester oder variabler Index in Klammern (indizierbare Operationen) angegeben werden, z.B: FGR(10) oder PAGE(X). Der Index kann auch mit einem Komma von der Feldgruppe abgetrennt werden.

Konstanten

für alphanumerische Werte in der Form: '*', 'EF' oder 'Falsche Taste'
als hexadezimale Werte in der Form: X'00', X'1DF0' oder X'FFFFFF'
für numerische Werte in der Form: 1 -10 123,45 oder 1000-

Je nach Version werden bei numerischen Konstanten unterschiedliche Dezimal- und Kolonnentrennzeichen verwendet:

Dezimalzeichen	deutsche Version ,	englische Version .
Kolonnentrennzeichen	deutsche Version .	englische Version ,
Beispiel:	1.234.567,89	1,234,567.89

= Zuweisen von Werten

```
EG OP F2 (SV)
```

```
EG   Feldname des Ergebnisfeldes
OP   '='
F2   Feld oder Konstante
SV   Runden 'H' oder 'ROUnded' bei num. Feldern.
```

Beispiele:

```
X = 1
A = B ROUNDED
STERN = '****'
```

Zweck:

In das Ergebnisfeld kann eine Konstante oder der Inhalt einer Konstanten übertragen werden. Sind EG und F2 beide numerisch, so wird EG vor der Übertragung gelöscht. Sind EG und F2 beide alphamerisch, so erfolgt eine Übertragung wie bei MOVE-LEFT (su.). Haben EG und F2 unterschiedliche Typen, d.h ein Operand ist alphamerisch und der andere numerisch, so erfolgt eine Übertragung wie bei MOVE-RIGHT (s.u.).

+ Addition

```
EG = F1 OP F2 (SV)
```

```
EG   Feldname des Ergebnisfeldes
OP   '+'
F1   Feld oder Konstante
F2   Feld oder Konstante
SV   'H' oder 'ROUnded' zum Runden
```

Beispiele:

```
C = A + B
X = Y + 5 ROUNDED
```

Zweck:

In das Ergebnisfeld wird die Summe von F1 und F2 übertragen. Das Ergebnis kann gerundet werden.

- Subtraktion

```
EG = F1 OP F2 (SV)
```

EG Feldname des Ergebnisfeldes
OP '-'
F1 Feld oder Konstante
F2 Feld oder Konstante
SV 'H' oder 'ROUnded' zum Runden

Beispiele:

```
C = A - B
X = Y - 5 ROUNDED
```

Zweck:

In das Ergebnisfeld wird die Differenz von F1 zu F2 übertragen. Das Ergebnis kann gerundet werden.

* Multiplikation

```
EG = F1 OP F2 (SV)
```

EG Feldname des Ergebnisfeldes
OP '*'
F1 Feld oder Konstante
F2 Feld oder Konstante
SV 'H' oder 'ROUnded' zum Runden

Beispiele:

```
C = A * B
X = Y * 5 ROUNDED
```

Zweck:

In das Ergebnisfeld wird das Produkt von F1 und F2 übertragen. Das Ergebnis kann gerundet werden.

/ Division

```
EG = F1 OP F2 (SV)
```

```
EG   Feldname des Ergebnisfeldes
OP   '/'
F1   Feld oder Konstante
F2   Feld oder Konstante
SV   'H' oder 'ROUnded' zum Runden
```

Beispiele:

```
C = A / B
X = Y / 5 ROUNDED
```

Zweck:

In das Ergebnisfeld wird der Quotient aus F1 und F2 übertragen. F2 darf nicht 0 sein. Das Ergebnis kann gerundet werden.

Bei einer Divison durch 0 wird eine Fehlermeldung (DEBUG) bei der Ausführung am Bildschirm angezeigt.

ACCEPT Datensatz direkt lesen

identisch mit CHAIN (s.u.)

AFOOT Mittelwert einer Feldgruppe rechnen

```
OP F2 EG (SV)
```

```
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
```

Beispiele:

```
AFOOT   FG1 MW
AVERAGE FG1 MW ROUNDED
```

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(1)    125,00
                              FG1(2)     75,00
                              FG1(3)     85,00
                              FG1(4)      0,00
                              FG1(5)      0,00
                              FG1(6)      0,00

Inhalt des Feldes MW
nach Ausführung der Instruktion:        95,00
```

AVERAGE Mittelwert einer Feldgruppe rechnen

identisch mit AFOOT (s.o.)

BEGSR Beginn Unterprogramm

F1 OP

F1 Name des Unterprogramms
OP BEGSR muss eingetragen werden

Beispiel:

```
UPRO BEGSR
```

Zweck:

Anfang eines Unterprogramms

Beschreibung:

Jedes Unterprogramm muss mit dieser Operation beginnen. F1 enthält den Namen des Unterprogramms. Alle müssen am Ende der Procedure Division codiert sein. Unterprogramme werden mit EXSR oder PERFORM aufgerufen.

BREAK Eine Schleife beenden

OP (SV)

OP BREAK muss eingetragen werden
SV All, um geschachtelte Schleifen zu verlassen

Beispiel:

```
DO FROM 1 TO WERT WITH I
   :
   IF REST <= 0
      BREAK
   ENDIF
   :
ENDDO
```

Zweck:

Eine DO-, DO UNTIL- oder DO WHILE-Schleife soll beendet werden.

Durch den Befehl BREAK wird eine DO-Schleife unabhängig von der Schleifenbedingung sofort beendet. BREAK verzweigt hinter 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 verzweigt wird.

CHAIN Satz wahlfrei lesen

F1 OP FN (SV)

OP CHAIN oder ACCEPT ist einzutragen
F1 Feldname des Schlüssels
FN Dateiname
SV Service: 'U', 'UPDate', 'C', 'CHEck','P'

Beispiele:

```
KEY CHAIN DATEI
KDNR ACCEPT KUNDEN UPDATE
```

Zweck:

Ein Satz einer Datei soll wahlfrei gelesen werden.

Beschreibung:

Mit dieser Operation wird ein Satz der in F2 genannten Randomdatei mit dem in F1 genannten Schlüssel gelesen. Die Länge des Schlüssels muss mit der in der FILES-Division angegebenen Schlüssellänge übereinstimmen.

Wird kein Satz gefunden, so wird 'NF' für not found im internen Feld CPGFRC übergeben.

Ein 'C' oder 'CHEck' in SV bedeutet, dass der Satz nur auf Verfügbarkeit geprüft wird. In diesem Falle werden keine Daten gelesen.

'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.

Achtung: 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.

CHANG(E) Feldinhalte austauschen

F1 OP F2

OP CHANG oder CHANGE muss eingetragen werden
F1 Feldname
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 enthält den Wert von FELD1. Sind FELD1 und FELD2 überlagert, dann ist das Ergebnis unvorhersehbar.

```
Sonderfall: FELD CHANGE FELD.   * Nach der Ausführung hat FELD den
                                * Wert x'00'.
```

CHECK Dateistatus prüfen

CHECK-VAR Dateistatus prüfen variabel

OP FN

OP CHECK oder CHECK-VAR muss eingetragen werden
FN Dateiname bei CHECK oder Alphafeld bei CHECK-VAR

Beispiel:

```
CHECK DATEI.
CHECK-VAR FELD.
```

Zweck:

Der Status einer Datei (open, closed) soll geprüft werden, bzw. ob die Datei in der FCT definiert ist.

Beschreibung:

Bei CHECK wird bei FN der Name der zu prüfenden Datei eingetragen und bei CHECK-VAR der Name eines Alphafeldes, in dem der Dateiname variabel enthalten ist.

Als Ergebnis der CHECK-Operation werden in CPGFRC die Informationen zum Dateistatus übergeben. Diese sind:

```
'  '  die Datei ist offen.
'NO'  die Datei nicht offen.
'NF'  die Datei wurde nicht gefunden. Im CICS ist die Datei nicht
      in der FCT und im Batch weder im Hauptprogramm noch von ei-
      em Modul eröffnet.
'OD'  Im CICS: Die Datei ist offen, aber disabled.
```

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.

CLEAR Löschen Inhalte aller Datenfelder

OP

OP CLEAR muss eingetragen werden

Beispiel:

```
CLEAR
```

Zweck:

CLEAR löscht die Inhalte aller Datenfelder des Programms

Beschreibung:

alle Alphafelder und -feldgruppen werden mit Blanks gefüllt und alle numerischen Felder und -feldgruppen werden auf 0 gesetzt.

CLOSE Datei abschließen

OP FN

OP CLOSE muss eingetragen werden
FN Dateiname

Beispiel:

```
CLOSE DATEI.
```

Zweck:

Eine Datei soll abgeschlossen werden.

Beschreibung:

Die explizite CLOSE-Operation schließt eine Datei ab. FN bezeichnet die abzuschließende Datei. Der Dateistatus kann im internen Feld CPGFRC abgefragt werden. Dabei steht 'NC' für 'not closed' und 'NF' für 'nicht in der FCT gefunden'.

COM-REG Communication Region

COMRG Communication Region

OP EG

OP COMRG oder COM-REG muss eingetragen werden
EG Name eines 32-stelligen Alphafeldes

Beispiele:

```
COM-REG FELD.
COMRG FELD.
```

Zweck:

System-Informationen werden dem Programm zur Verfügung gestellt.

Beschreibung:

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	Trans-Id
11 - 12	Anzahl Bildschirmzeilen Default (gepackt)
13 - 14	Anzahl Bildschirmspalten Default (gepackt)
15 - 16	Anzahl Bildschirmzeilen Alternate (gepackt)
17 - 18	Anzahl Bildschirmspalten 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: Es wird nicht Übersetzt 'T' = UCTRAN Transaktion (ab CICS 2.2) 'U' = UCTRAN ein: Übersetzen in Großbuchstaben
28 - 32	Reserviert

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.       *  Alles
       OPID         3.       *  Operator-Id
       CURSOR       5 0.     *  Cursorposition
       TRANID       4.       *  Transid
       DZEILE       3 0.     *  Anzahl Bildschirmzeilen default
       DSPALT       3 0.     *  Anzahl Bildschirmspalten default
       AZEILE       3 0.     *  Anzahl Bildschirmzeilen alternate
       ASPALT       3 0.     *  Anzahl Bildschirmspalten alternate
       TSKNO        5 0.     *  Task nummer
       UCTRNB       1.       *  UCTRAN byte
       FILLER       4.       *  Reserviert
       UCTRNE       1.       *  UCTRAN Byte erweitert
       REST         5.       *  Reserviert
 -C
       COMRG FELD
```

COMPUTE Berechnung von Formeln

Mit dem Befehl COMPUTE wird der Formula Translator aufgerufen. Der Formelübersetzer erlaubt die Berechnung von Formeln und mathematischen Funktionen.

Beispiele:

```
COMPUTE   Y = ( A + B ) * ( C - D )
COMPUTE   A = D ^ 2 * 3,14159 / 4
COMPUTE   Y = SIN(X)
COMPUTE   WINKEL = ARCTAN( HOEHE / BREITE )
COMPUTE   POTENZ = X ^ Y
COMPUTE   Z = LOG(X) * SQRT(Y)
compute   x = exp((sin 30)_(2-3*tan 45)*ln 10)/5
compute   x = ((1-2^-1)/((3-4^1)))/((((10-2^3))))
compute   x = -10^3
compute   x = -2^((-sin 30)^-1)
compute   tmp3  =  2,674*10^11 * 800^-0,5 * 35^-4,93
```

Bei den Berechnungen wird intern die Gleitpunktdarstellung in doppelter Genauigkeit (15 gültige Ziffern auch für Zwischnergebnisse) verwendet. Neben den Grundrechenarten (+, -, *, /) sind Funktionen sind in einer Mathematical Library verfügbar. Diese enthält z.Zt. folgende Funktionen:

^	Potzenz (x hoch y)
ARCCOS	Arcus Cosinus
ARCSIN	Arcus Sinus
ARCTAN	Arcus Tangens
COS	Cosinus
COT	Cotangens
EXP	Exponentialfunktion (e hoch x)
LN	Logarithmus naturalis
LOG	Logarithmus
SIN	Sinus
SQRT	Quadratwurzel
TAN	Tangens
10^	Potenz zur Basis 10

Bei den Winkelfunktionen werden die Argumente in Grad vorgegeben. Bei den Umkehrfunktionen ARC... wird das Ergebnis in Grad zurückgegeben. Sind Werte im Bogenmaß angegeben, bzw. soll das Ergebnis im Bogenmaß ermittelt werden, so erfolgt die Umrechnung leicht mit folgenden COMPUTE-Statements:

```
COMPUTE   BOGEN = GRAD  * PI / 180
COMPUTE   GRAD  = BOGEN * 180 / PI
```

Die Funktionen dürfen nicht als Variablen-Namen benutzt werden. Bei den Operanden sind keine indizierten Variablen unterstützt. Die maximale Schachtelungstiefe bei Klammern ist 10. Die mathematischen Syntaxregeln müssen eingehalten werden. Ist der Ausdruck insgesamt zu lang, so erfolgt ebenfalls eine Fehlermeldung und das Statement muss gegebenenfalls in 2 COMPUTE-Operationen geteilt werden.

Bis auf Blanks vor und nach dem '=' Zeichen ist es nicht erforderlich, die Operanden mit Leerstellen zu trennen.

Die Berechnung der mathematischen Funktionen erfolgt intern mit Gleitpunktoperationen doppelter Genauigkeit (ca. 15 gültige Ziffern). Durch geeignete Wahl der Größe des Ergebnisfeldes wird damit eine maximale Genauigkeit erreicht.

Gleitpunktoperationen werden auch bei Multiplikation und Division benutzt, um mehr Stellen beim Ergebnis zu erzielen. Außerdem wird das Ergebnis intern gerundet. Hierdurch kann bei COMPUTE das Ergebnis genauer sein als bei konventioneller Operationsfolge, z.B:

```
COMPUTE X = ( 1 / 3 ) * 3.     * Ergebnis X = 1
X = 1 / 3.                     * und
X = X * 3.                     * X = 0,999999 (wenn X 6 Dezimalen hat)
```

Bei einem Überlauf (Ergebnis > 15 Vorkommastellen) wird die Verarbeitung abgebrochen. Bei Aufruf am Bildschirm erscheint hier die Testhilfe (DEBUG).

Bei COMPUTE können auch Feldgruppen ohne Index verarbeitet werden. Dadurch wird mit einem Befehl eine ganze Gruppe von Berechnungen ausgeführt. Werden mehrere Feldgruppen als Operanden bei COMPUTE verwendet, so sollten alle die gleiche Anzahl Elemente haben, ansonsten wird die Gruppe nur für die kleinste Anzahl von Elementen in einer der beteiligten Feldgruppen berechnet.

CONT(INUE) Fortsetzen einer Schleife

OP

OP CONTINUE oder CONT muss eingetragen werden

Beispiel:

```
DO FROM 1 TO WERT
   :
   IF FELD = 0
      CONTINUE
   ENDIF
   EDIT PAGE
ENDDO
```

Zweck:

Eine DO-, DO UNTIL- oder DO WHILE-Schleife soll mit dem nächsten Schleifendurchlauf fortgesetzt werden.

Die Schleife wird entsprechend der Schleifenbedingung durchlaufen. Im Fall FELD = 0 wird die Operation EDIT jedoch nicht ausgeführt. CONTINUE verzweigt zurück zum zugehörigen DO-Befehl.

CONVERT Konvertieren eines alphanumerischen Feldes

CONVT Konvertieren eines alphanumerischen Feldes

OP (F2 INTO) EG (SV)

OP CONVERT oder CONVT muss eingetragen werden
F2 Name des zu konvertierenden Feldes
EG Ergebnisfeld (INTO sollte codiert werden)
SV CHAracter, DATe, HEX, LOW, SEConds, TIMe, U, Year

Beispiele:

```
CONVERT A1
CONVERT A1 INTO A2  HEXADECIMAL
```

Zweck:

Der Inhalt eines alphanumerischen Feldes soll konvertiert werden. Bei der Konvertierung von Zeit- und Datums-Informationen sind auch numerische Operanden erlaubt.

Beschreibung:

CONVERT A1 übersetzt A1 von 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, U 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	Format: (0)TTMMJJ
alpha	8	num.	8,0 oder 9,0	Format: (0)TTMMJJJJ
alpha	8			Format: TT.MM.JJ
alpha	10			Format: TT.MM.JJJJ

DAT vertauscht Jahr und Tag, so dass die Jahreszahl nach der Operation vorne im Feld steht.

U Konvertiert ein Datum vom ISO-Format jjjjmmtt in das Standardformat tt.mm.jjjj. Das Ergebnisfeld muss 8 oder 10 Stellen alpha sein und Faktor 2 numerisch (8,0 Stellen) oder alpha mit 8 Stellen.

Y für Year wird benötigt,falls die Jahreszahl in den ersten vier Stellen des Feldes steht. YEAR ist die Umkehrfunktion zu DATE. (Bei zweistelligen Jahreszahlen genügt die Servicefunktion DATe für beide Konvertierungsrichtungen).

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 aber 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.

Je nach Service SV gilt für die Eintragungen in F2 und EG:

SV	F2	EG	Kommentar
	A	A	
C	A	A	F2 und EG dürfen nicht überlagert sein
D	A	A	F2 und EG jeweils 6, 8 oder 10 Stellen – oder
D	N	N	F2 und EG jeweis 6,0/7,0 oder 8,0/9,0 Stellen
L	A	A	
S	N	N	F2 mit 5,0 bis 9,0 Stellen
T	N	N	F2 mit 5,0 bis 9,0 Stellen
U	N	A	F2 mit 8,0 und EG mit 8 oder 10 Stellen
X	A	A	F2 und EG dürfen nicht überlagert sein
Y	A	A	F2 und EG jeweils 6, 8 oder 10 Stellen – oder
Y	N	N	F2 und EG jeweis 6,0/7,0 oder 8,0/9,0 Stellen

A = alpha, N = numerisch

DEBUG Testhilfe-Funktion

OP (F2)

OP 'DEBUG' muss eingetragen werden
F2 ON oder OFF um DEBUG ein oder auszuschalten.

Beispiele:

```
DEBUG
:
DEBUG ON
:
DEBUG OFF
```

Zweck:

1. Ein Test-Bild wird am Bildschirm angezeigt.

2. Die Möglichkeit, mit DEBUG zu testen, wird für einen Programmabschnitt ein- bzw. ausgeschaltet.

DELC Zeichen entfernen

OP F2 EG (SV)

OP DELC muss eingetragen werden
F2 das oder die zu entfernenden Zeichen
EG Name eines Alphafeldes oder -feldgruppenelements
    oder Name einer alphanumerischen Feldgruppe
SV 'A' für alternative Feldgruppenverarbeitung

Beispiele:

```
DELC '0'  FELD
DELC X'FF' INPUT
DELC '*' FG A
DELC 'AEIOUaeiou' WORT
```

Zweck:

Löschen von Zeichen aus einem alphanumerischen Feld.

Beschreibung:

Mit dieser Operation kann man Zeichen aus einem alphanumerischen Feld löschen. Dabei werden die folgenden Zeichen nach links verschoben und die letzten Stelle werden mit Blank(s) gefüllt.

Nach dem Operationscode DELC werden die zu entfernenden Zeichen als Konstante oder in einem Alphafeld angegeben.Dahinter folgt der Name des zu bearbeitenden Feldes.

```
Beispiel :                  DELC '*' FELD

Feldinhalt vorher  :       'A*B**C'
Feldinhalt nachher :       'ABC   '
```

Bei Feldgruppen wird die gesamte Feldgruppe als ein zusammenhängender Bereich angesehen, d.h. wird ein Zeichen entfernt, dann werden alle folgenden Zeichen auch elementübergreifend zusammengeschoben. Wird jedoch der Service 'A' angegeben, dann wird die Feldgruppe elemnentweise verarbeitet, d.h. hierbei erfolgt das Verschieben jeweils nur innerhalb eines Feldgruppenelementes.

Ab Release 2.6 können mit einem DELC-Befehl mehrere Zeichen aus einem alphanumerischen Feld entfernt werden.

```
Beispiel :                  DELC 'ACDEF' FELD

Feldinhalt vorher  :       'A*B**C'
Feldinhalt nachher :       '*B**  '
```

DELET(E) Satz von einer Datei löschen

(F1) OP FN

F1 Schlüsselfeld, das den zu löschenden Satz angibt
OP DELET oder DELETE muss eingetragen werden
FN Name der Datei, auf der gelöscht wird

Beispiele:

```
KEY  DELET  CPGWRK
DELETE KUNDEN
```

Zweck:

Löschen eines Satzes in einem Datenbestand.

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 Name der Datei, von der der Satz gelöscht werden soll.

DEQ Programmteil entriegeln

(F1) OP

F1 Identifier, gleicher Name wie beim ENQ
OP DEQ muss eingetragen werden

Beispiele:

```
ENQ
:                              zu sperrender Programmteil
DEQ
```

Beschreibung:

DEQ gibt einen mit ENQ gesperrten Programmteil wieder frei.

DISPLAY Konsolmeldung ausgeben

identisch mit DSPLY (s. u.)

DO Befehlsfolge ausführen

OP (SV) (DY) (F1) (DY) (F2) (DY) (EG)

```
OP   DO muss eingetragen werden
SV   LOOP bei Endlosschleifen
DY   Dummywort: 'FROM' bei F1
F1   Startwert, nicht bei DO LOOP
DY   Dummywort: 'TO', 'TIMES' bei F2
F2   Endwert, Anzahl Durchläufe, nicht bei DO LOOP
DY   Dummywort: 'WITH' bei EG
EG   Index. Name eines numerischen Feldes mit 0 Dezimalstellen
```

Beispiele:

```
DO.                              * Befehlsfolge ausführen
DO 10.                           * 10 mal ausführen
DO 10 TIMES.                     * 10 mal
DO 5 TIMES WITH I.               * 5 mal mit Index I
DO FROM X TO Y WITH I.           * Anfangswert X, Endwert Y
DO LOOP.                         * Endlos-Schleife
DO LOOP WITH I.                  * Endlos-Schleife mit Index I
```

Zweck:

Eine Folge von Anweisungen soll 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 oder Literal ohne Dezimalen sein, das den Anfangswert des Schleifenzählers enthält.

F2 kann ein numerisches Feld oder Literal ohne Dezimalen sein, das den Endwert des Schleifenzählers enthält. Ist F2 nicht angegeben, so ist der Begrenzungswert 1.

In EG kann ein numerisches Feld ohne Dezimalstellen eingetragen werden, in das während der Ausführung der aktuelle Index gestellt wird.

In SV wird 'LOOP' angegeben, wenn die Schleife endlos sein soll.

Die zu verarbeitende Befehlsfolge muss durch ein END- oder ENDDO – Statement abgeschlossen werden.

DO und IF können bis zu 20 Stufen tief verschachtelt werden.

DO UNTIL Befehlsfolge ausführen bis

OP (F1) (OC) (F2) (SV)

OP DO UNTIL oder DO UNTIL-DAT(I)
F1 erstes Vergleichsfeld
OC Operator (s.u., Tabelle der Operatoren)
F2 zweites Vergleichsfeld
EG Index. Name eines numerischen Feldes mit 0 Dezimalstellen
SV Boolesche Operatoren AND und OR

Beispiele:

```
DO UNTIL X = 0.                 * bis X = 0 und
         Y = 0.                 * bis Y = 0
DO UNTIL-DAT D1 < D2 AND
   UNTIL-DATI D1 > '991231' OR
   D3 > D4
```

Zweck:

Eine Folge von Anweisungen soll wiederholt ausgeführt werden.

Beschreibung:

Bei DO UNTIL erfolgt die Abfrage der Schleifenkriterien vor dem Schleifendurchlauf.

F1 und F2 können beliebige numerische oder alphanumerische Felder, Feldgruppenelemente oder Konstanten sein und müssen vom gleichen Typ sein.

Die Befehlsfolge muss durch END oder ENDDO beendet werden.

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

DO und IF können bis zu 20 Stufen tief verschachtelt werden.

Bei den booleschen Operatoren AND und OR kann die Folgezeile mit DO UNTIL, mit UNTIL, oder nur mit dem Vergleich fortgesetzt werden.

DO UNTIL-DATe und DO UNTIL-DATI erlauben die Abfrage von Datumfeldern im Standard- (-DAT) oder im ISO-Format (-DATI). Mit AND und OR können Datumsabfragen miteinander und mit 'normalen' DO-UNTIL-Abfragen verknüpft werden.

DO WHILE Befehlsfolge ausführen solange

OP (F1) (OC) (F2) (SV)

OP DO WHILE, DO WHILE-DAT oder DO WHILE-DATI
F1 erstes Vergleichsfeld
OC Operator (s.u., Tabelle der Operatoren)
F2 zweites Vergleichsfeld
EG Index. Name eines numerischen Feldes mit 0 Dezimalstellen
SV Service: 'LOOP'
SV Boolesche Operatoren AND und OR bei DO WHILE oder DO UNTIL

Beispiele:

```
DO WHILE A = B OR                * solange A = B oder
DO WHILE C = D AND               * solange C = D und
   WHILE E = F OR                * solange E = F
         A >= FGA(I)
DO WHILE A EQUAL B.
DO WHILE FGN(3) IS EQUAL ZAHL.
```

Zweck:

Eine Folge von Anweisungen soll auf Grund eines Vergleichs ein- oder mehrmals ausgeführt werden.

Beschreibung:

Bei DO WHILE erfolgt die Abfrage der Schleifenkriterien vor dem Schleifendurchlauf.

F1 und F2 können beliebige numerische oder alphanumerische Felder, Feldgruppenelemente oder Konstanten sein und müssen vom gleichen Typ sein.

Die Befehlsfolge muss durch END oder ENDDO beendet werden.

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

DO und IF können bis zu 20 Stufen tief verschachtelt werden.

Bei den booleschen Operatoren AND und OR kann die Folgezeile mit DO WHILE, mit WHILE oder nur mit dem Vergleich fortgesetzt werden. DO WHILE-DATe und DO WHILE-DATI erlauben die Abfrage von Datumfeldern im Standard- (-DAT) oder im ISO-Format (-DATI). Mit AND und OR können Datumsabfragen miteinander und mit den 'normalen' DO-

DSPLY Ausgabe auf die Operator-Konsole

F1 OP	ONLINE und BATCH
(F1) OP (EG)	nur für BATCH

F1 Name des Feldes oder Konstanten, das die Nachricht enthält
OP DSPLY oder DISPLAY muss eingetragen werden
EG Name des Feldes, das eine Nachricht/Antwort enthält

Beispiele :

```
MSG  DSPLY
'Fehler 17' DSPLY
MSG1 DISPLAY RESP
'Bitte Jahr eingeben' DISPLAY JAHR
123,45 DSPLY
```

Zweck:

Ausgabe von Nachrichten auf der 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 F1 kann ein alphanumerisches oder numerisches Feld oder eine alphanumerische oder numerische Konstante sein (im Batch maximal 64 Bytes).

Im Batch wartet DSPLY auf die Eingabe des Operators, wenn bei EG ein Feld angegeben wurde. Die 'Antwort' der Konsole wird im Ergebnisfeld an das Programm übergeben.

Das Ergebnis EG kann ebenfalls ein alphanumerisches oder numerisches Feld sein.

DUMP Speicherauszug

OP (F2)

OP DUMP muss eingetragen werden
F2 ON, OFF oder Dump-Code

Beispiele :

```
DUMP
DUMP ON
DUMP OFF
DUMP HUGO
```

Zweck:

Steuern Handling und Speicherauszug bei einem Programmabbruch

Beschreibung:

Mit den Parametern 'ON' oder 'OFF' wird gesteuert, wie sich QPG im CICS bei einem Programmabbruch innerhalb der QPG-Ausführung verhält. Bei ON wird ein CICS-Transaction-Dump erzeugt. Bei OFF wird kein DUMP erzeugt, sondern es wird die Testhilfe DEBUG aufgerufen und im Testbild wird das Statement angezeigt, bei dem der Programmabbruch aufgetreten ist. Die Standardeinstellung im CICS ist OFF. Siehe auch Testhilfe Debug.

Ohne F2 oder mit Angabe eines Dump-Codes wird ein Haupspeicherauszug erstellt. Im CICS ist dabei Voraussetzung, dass der Dumpbereich offen ist.

Tip:

Bei einem Fehler in einem HL1-Modul wird Debug bei Statement EXHM aufgerufen. Falls hiermit der Fehler noch nicht ausreichend ermittelt werden kann, sollte vor EXHM mit DUMP ON die Fehlerbehandlung wie bisher von CICS erfolgen und nach EXHM mit DUMP OFF wieder die QPG-Fehlerbehandlung aktiviert werden.

EDIT alphanumerisches Feld aufbereiten

OP EG (KW F2)

OP EDIT muss eingetragen werden
EG Name des aufzubereitenden Feldes
KW Dummywort: TYPE oder SEGMent
F2 Segment Name in Output Division

Beispiele:

```
EDIT  FELD.
EDIT  FG1(J).
EDIT  ZEILE TYPE POSTEN
EDIT  PAGE SEGMENT KOPF
EDIT  PAGE SEGM SUMME
EDIT  CPGCOM.          * Ausgabe Common Area (nur Command Level)
EDIT  CPGTCT.          * Ausgabe TCT User Area (nur im ESA Mode)
```

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. Wird SEGMent oder TYPE angegeben, so wird in der Ausgabe das Feld mit einer anderen Struktur aufbereitet.

Die Umwandlung wird mit einer Fehlermeldung abgebrochen, wenn das Feld kürzer ist als die maximale Ausgabeposition. Das gilt auch, wenn SEGMent oder TYPE angegeben ist.

Bei EDIT CPGTCT wird bei Ausführung im CICS die Länge der TCT User Area geprüft. Ist diese nicht vorhanden oder zu klein, so erfolgt ein Programmabbruch.

ELIM(INATE) Zeichen löschen und durch Blank ersetzen

OP F2 (DY) EG

OP ELIM oder ELIMINATE muss eingetragen werden
F2 zu entfernende(s) Zeichen
DY Dummywort FROM
EG Alphafeld / -feldgruppe / -feldgruppenelement

Beispiele:

```
ELIM '_'  FELD
ELIMINATE  X'00' FROM INPUT
ELIM '_-./+'  FELD
```

Zweck:

Löschen von Zeichen aus einem alphanumerischen Feld und Ersetzen durch Blank.

Beschreibung:

Mit dieser Operation können ein Zeichen oder mehrere in einem alphanumerischen Feld, einer Feldgruppe oder einem Feldgruppenelement durch Blank ersetzt werden. Die zu ersetzenden Zeichen werden entweder als Konstante oder in einem Alphafeld angegeben.

```
Beispiel :                  ELIM '*' FELD

Feldinhalt vorher  :       'A*B**C'
Feldinhalt nachher :       'A B  C'
```

Ab Release 2.6 können mit einem ELIM-Befehl mehrere Zeichen aus einem alphanumerischen Feld entfernt werden.

```
Beispiel :                   ELIM 'ACDEF' FELD

Feldinhalt vorher  :        'A*B**C'
Feldinhalt nachher :        ' *B** '
```

ELSE Kennzeichnung eines Programmblocks im IF-Befehl

OP

ELSE

Die Operation ELSE kennzeichnet innerhalb einer IF-Gruppe den Programmzweig, der durchlaufen wird, wenn die Bedingung in der IF – Operation nicht erfüllt ist.

END Ende eines Programmblocks

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 eines EVALUATE Blocks

identisch mit ENDEV (s. u.)

ENDDO Ende eines DO-Blocks

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.

ENDEV Ende eines EVALUATE Blocks

Die Operation ENDEV oder END-EVALUATE beendet eine EVALUATE Gruppe.

ENDIF Ende eines IF-Blocks

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.

ENDPR Programm beenden

ENDPR wird benutzt, um ein Programm vorzeitig zu beenden. Die Kontrolle und die Daten werden dann an das aufrufende Programm zurückgegeben.

ENDSR Ende Unterprogramm

OP

OP ENDSR muss eingetragen werden

Beispiel:

```
ENDSR
```

Zweck:

Beenden eines Unterprogramms

Beschreibung:

Das Programm verzweigt bei dieser Operation zurück zum aufrufenden Befehl EXSR bzw. PERFORM.

ENQ Programmteil sperren

OP (F2)

OP ENQ muss eingetragen werden
F2 Identifier: Name des Labels, bis zu dem gesperrt wird

Beispiele:

```
ENQ
:                              zu sperrender Programmteil
DEQ
```

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.)

EVALUATE Mehrfachalternative

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-Folgezeilen gekennzeichnet. Die Operanden für die Bedingungen können numerische oder alphanumerische Konstanten, Felder oder Feldgruppenelemente sein. Ist eine Bedingung erfüllt, werden die nachfolgenden Anweisungen bearbeitet. Danach ist die EVALUATE-Anweisung beendet und das Programm wird hinter 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. EVALUATE-Gruppen können bis zu 10 Stufen tief geschachtelt werden.

Beispiel:

```
-C
.
EVALUATE
   WHEN X = 1
      .                * 1. Fall
   WHEN X = 2
      .                * 2. Fall
   WHEN X = 3 OR
        X = 4 AND
        A = 'ABC'
      .                * 3. Fall
   WHEN-DAT VDAT < UDATE OR
   WHEN-DATI XDAT > CPGDAI AND
   WHEN A = 'XYZ'
      .                * 4. Fall
        EVALUATE
             WHEN KZ = 'A'
                .
             WHEN KZ = 'B'
                .
             WHEN OTHER
                .
        END-EVALUATE
   WHEN OTHER
      .                * sonst
END-EVALUATE
```

EXHM HL1-Modul ausführen

OP F2 (EG) (SV)

OP EXHM muss eingetragen werden
F2 Name eines HL1-Moduls
EG Name eines HL1-Datenkanals
SV I : Baugruppe initialisieren, T : I + PF-Tasten

Beispiele:

```
EXHM  HB0001
EXHM  HB0002  KANAL
EXHM  HB0003  T
EXHM  HB0004  KANAL I
```

Zweck:

Ein HL1-Baustein soll ausgeführt werden.

Beschreibung:

In F2 wird der Name eines Bausteins, der in der HL1-Tabelle enthalten sein muss, angegeben.

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 beschriebenen Felder in die private TWA (Transaction Work Area) des Bausteins übertragen. Nach dessen Ausführung werden die evtl. veränderten Feldinhalte wieder in die entsprechenden Felder des aufrufenden Programms zurück übertragen.

Ist in den Options des HL1-Moduls der Parameter SHAre eingetragen, dann erfolgt der Datenaustausch wie bei der Operation PROG über den Feldnamen. Da der Datenkanal auf 3936 Stellen beschränkt ist, empfiehlt es sich die Daten mit SHAre-Logik zu übertragen. SHAre und der Datenkanal in der Input Division sollten nicht kombiniert werden.

Mit dem Service 'I' oder INIT bewirkt man, dass der Ablauf hier fortgesetzt wird, wenn in einem Modul dialogorientiert die CLear-Taste betätigt wurde.

Service 'T' beinhaltet Service 'I'. Zusätzlich können Programmfunktionstasten abgefragt werden.

Bei EXHM ermöglicht die Option CLEAR die Abfrage der CLEAR-Taste und hat die gleiche Wirkung wie der Service 'I'. Ohne diese Option wird durch die CLEAR-Taste die aktuelle Transaktion im CICS beendet.

EXITD Andere Transaktion starten mit Datenübergabe

OP EG (SV)

OP EXITD muss eingetragen werden
EG Name einer Datenstruktur
SV 'T' für Uhrzeit (statt Zeitintervall)

Beispiel:

```
EXITD  STRUKT
```

Zweck:

Starten einer anderen Transaktion mit Datenübergabe. EXITD wird z.B. benutzt um Drucker-Tasks zu aktivieren.

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 Data-Division wie folgt beschrieben sein muss:

```
-D
  STRUKT 0 * 88.      * DS: kann bis zu 256 Stellen lang sein.
      TRANID  4.      * 1-4 Trans-Id der Folgetask
      TERMID  4.      * 5-8 Terminal-Id, an dem die Folgetask
                      * gestartet wird. Default : gleiches Ter-
                      * minal
      ZEIT    7 0.    * 9-12 entwd. Uhrzeit oder Intervall zum
                      * Zeitpunkt des Befehls. Gibt an, wann
                      * die Folgetask gestartet werden soll.
      TSNAME  8.      * 13-20 Temp. Storage Name. Falls Blank,
                      * wird der Name vom TP-Monitor vergeben.
      INTERN  4.      * 21-24 wird intern vom CPG gefüllt.
      DATEN  64.      * 25-88 zu übergebende Daten. Die Benut-
                      * zerdaten können max 232 Byte lang sein.
```

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, z.B. 20 für einen Start der Folgetask nach 20 Sekunden oder 230 für ein Intervall von 2 Minuten und 30 Sekunden.

Das interne Feld CPGFRC wird mit 'EF' gefüllt, wenn entweder die auszuführende Task oder das angesprochene Terminal nicht in den entsprechenden CICS-Tabellen definiert ist, ansonsten wird CPGFRC mit ' ' gefüllt.

Die Folgetask kann die übergebenen Daten über eine READ-Operation auf '$CPG', einer CPG-internen Temporary Storage Queue, lesen. Wenn kein READ auf $CPG erfolgt, so werden diese Daten automatisch beim Ende der Task gelöscht. Beispiel:

```
-I.
   FILE $CPG.
                   1  54 DATEN
-C.
   READ $CPG
   .
```

Unter $CPG wird eine Pseudo-TS-Queue gelesen; ist diese nicht vorhanden, wird der Schalter EF gesetzt.

EXITI Anderes Programm über Intervall Control aufrufen

(F1) OP F2 (SV)

OP	EXITI muss eingetragen werden
F1	Terminal-Id an dem die Task gestartet werden soll
F2	Transaktion in Hochkommata oder als Variable
SV	N wenn die aktuelle Task nicht verlassen werden soll

Beispiele:

```
EXITI 'QTF '
EXITI TRID
'DR01' EXITI TRID N
```

Zweck:

Aufrufen einer anderen Trans-Id mit Intervall Control.

Beschreibung:

F1 kann eine Terminal-Id oder 'NONE' als Konstante oder als Feldinhalt enthalten, wenn die Transaktion an einem anderen Bildschirm oder im Hintergrund gestartet werden soll.

F2 enthält die Transaktion der aufzurufenden Trans-Id 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.

Mit Service 'N' wird die aktuelle Task nicht beendet. Diese Angabe erfolgt z.B. beim Starten an einem anderen Bildschirm oder im Hintergrund.

EXIT-TRANS Aufruf der nächsten Transaktion

EXITT Aufruf der nächsten Transaktion

OP F2

OP EXITT oder EXIT-TRANS muss eingetragen werden
F2 Transaktion in Hochkomma

Beispiel:

```
EXITT  'TST1'
EXITT  TRID
EXITT  ' '.          * Return to CICS
```

Zweck: Aufruf nächste Transaktion

Beschreibung:

Die Operation EXITT startet die nächste CICS Transaktion mit der angegebenen Trans-Id. Modifizierte Bildschirmfelder können danach mit der Operation MAP übertragen werden.

Wird ' ' als Trans-Id angegeben, dann wird die Kontrolle an CICS zurückgegeben.

EXSR Unterprogramm ausführen

OP F2

OP EXSR oder PERFORM muss eingetragen werden
F2 Name des Unterprogramms

Beispiel:

```
EXSR UPRO
PERFORM RECHNEN
```

Zweck: Aufruf eines Unterprogramms

Beschreibung:

Mit EXSR oder PERFORM verzweigt das Programm in die in F2 angegebene Unterroutine, die am Ende der Procedure Division programmiert sein muss.

EXSR ist mit PROG vergleichbar, nur wird hier eine interne Unterroutine aufgerufen. Dadurch ist die Performance besser.

FILL Feld mit einem Zeichen füllen

OP F2 (DY) EG

OP FILL muss eingetragen werden
F2 Füllzeichen als Konstante oder Feld
DY Dummywort TO oder INTO
EG Name eines alphanumerischen Feldes

Beispiele:

```
FILL  ' '    PAGE
FILL  X'00'  TO INFO
```

Zweck:

Ein alphanumerisches Feld oder eine Feldgruppe sollen mit einem Zeichen gefüllt werden.

Beschreibung:

Mit der Operation FILL kann ein alphanumerisches Feld mit jedem beliebigen Zeichen gefüllt werden.

F2 enthält das Füllzeichen entweder als einstellige Konstante oder als erste Stelle eines Feldes.

Beispiel: FILL '*' FELD

Feldinhalt vorher	'123 '
Feldinhalt nachher	'******'

FIND Durchsuchen einer Tabelle

(F1) OP F2

F1 Suchargument
OP FIND muss eingetragen werden
F2 (vierstelliger) Name der Tabelle

Beispiele:

```
K  FIND  TAB1
PLZ2  FIND TAB2
```

Zweck:

Eine (extern erstellte) Tabelle wird nach einem Feldinhalt durchsucht.

Beschreibung:

Voraussetzung ist, dass eine Tabelle angelegt wurde, die mit FIND verarbeitet werden kann. Das Erstellen einer solchen Tabelle mit QTS ist im Handbuch der CPG3-Serviceprogramme beschrieben.

F1 enthält den Namen eines Feldes, das die zu durchsuchende Spalte der Tabelle angibt und nach dessen Inhalt diese Spalte durchsucht wird. Wird F1 nicht angegeben, so werden alle Einträge der Tabelle sequentiell verarbeitet.

In F2 steht der vierstellige Name der Tabelle.

Je nachdem ob das Suchargument gefunden wurde oder nicht, wird im Feld CPGFRC der Status der Operation gesetzt:

- `' '` wenn das Argument gefunden wurde.
- `'EF'` wenn das Ende der Tabelle erreicht wurde, d.h. das Argument wurde nicht gefunden.

GETHS Get higher Storage

GETHS

Zweck:

In einem QPG-Dataset sollen die Feldinhalte so wiederhergestellt werden, wie sie bei Aufruf vom übergeordneten Programm übergeben wurden.

Beschreibung:

GETHS wird in Datasets eingesetzt. Bei der Programmierung einer logischen Datei in einem Dataset kann die Schwierigkeit auftreten, dass Feldinhalte, die von einem übergeordneten Programm übergeben wurden, durch Lese-Operationen überschrieben werden, z.B. beim CHAIN vor einem Update.

Mit GETHS ist es möglich, in einem QPG-Dataset die Update-Funktion auch dann zu realisieren, wenn nur ein Teil der Felder vom aufrufenden Programm übergeben wurde.

Die Operation holt jeweils die Inhalte aller Felder vom übergeordneten Programm ab, die dort definiert sind und stellt damit deren Werte auf den Inhalt zurück, wie er bei Start des Datasets vorhanden war.

HTMLI Erneutes Übertragen der Daten im Intra-/Internet

OP

OP 'HTMLI' muss eingetragen werden

Beispiele:

```
HTMLI.                                  * Einlesen der Daten
```

Zweck:

Es sollen die Daten vom Internet oder Intranet eingelesen werden.

Beschreibung:

Mit HTMLI werden die Daten vom Inter-/Intranet erneut übertragen. Bei Programmen, die UPDATes auf Dateien durchführen, kann so der Datensatz mit CHAIN für Update gelesen und anschließend die Änderungen mit HTMLI vom Inter-/Intranet übertragen und mit UPDATe in der Datei gespeichert werden.

Bei HTMLI ist die OPTIONS HTML erforderlich.

HTMLO Ausgabe einer NetPage-Maske im Intranet oder Internet

OP (F2)

OP 'HTMLO' muss eingetragen werden
F2 ein NetPage-Name kann eingetragen werden

Beispiele:

```
HTMLO INVENTUR.                         * Ausgabe der Map INVENTUR
CPGHTM = VARMAP.                        * Zuordnen var. Mapname
HTMLO.                                  * Ausgabe der var. Map
```

Zweck:

Es soll eine Maske im Internet oder Intranet ausgegeben werden.

Beschreibung:

HTMLO erlaubt den direkten Aufruf von NetPage-Masken. Wird HTMLO in einem Unterprogramm aufgerufen, dann erfolgt kein Rücksprung mehr in das rufende Programm, sondern die NetPage-Maske wird mit Daten des Unterprogramms gefüllt. Wird HTMLO ohne Mapnamen angegeben, so kann ein variabler NetPage–Name benutzt werden, indem dieser vorher im internen Feld CPGHTM bereitgestellt wird.

Bei HTMLO ist die OPTIONS HTML erforderlich.

Im Mapnamen kann ein $ für den Sprachencode angeben werden. Das $ wird bei der Ausführung durch den Sprachencode ersetzt. Die Sprache ist in der Installation angegeben, kann aber durch das interne Feld CPGMPL geändert werden. Z.Zt. sind die Werte 'D' (deutsch) und 'E' (englisch) unterstützt.

IF Wenn - Abfrage

```
OP F1 (DY) OK F2 (DY) (SV)
```

```
OP   'IF' muss eingetragen werden
F1   erstes Vergleichsfeld
OC   Operator, vgl. unten : Tabelle der Operatoren
F2   zweites Vergleichsfeld
DY   Dummyworte ( wie z.B. IS, THAN )
SV   Boolesche Operatoren  AND  oder  OR
```

Beispiele:

```
IF A = B.                          * wenn A = B
IF A GT B  AND
IF SUMME > FG(IND) OR
   SUMME < 10000
IF A IS <= B.
```

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. 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 TYPE = 'KREIS'
   PI = 3,14.
   X = D * PI.
ELSE
   X = D * D
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 gleich	F2
LE	<=	=<	LESS EQUAL	F1	kleiner gleich	F2

In Verbindung mit den booleschen Operatoren AND und OR gelten folgende Syntaxregeln:

Zu einer logischen Verknüpfung von IFs gehört immer nur ein ENDIF. OR und AND stehen immer hinter dem IF-Statement.

Beispiele:

```
IF A = B  AND.                          *  Und-Verknüpfung
IF B > C
   EDIT INFO1
END
IF KDNR > '     '  OR                   *  Oder-Verknüpfung
IF SUMME > 0  OR
IF X = 0
   EDIT INFO2
ENDIF
IF A > B  AND                           *  Und-/Oder-Verknüpfung
IF A > C  OR
IF A = B  AND
   A = C  OR
   A = D  AND
   C = 0
    EDIT INFO3
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. EDIT wird ausgeführt, wenn eine der beiden Und-Verknüpfungen wahr ist. IF braucht in Fortsetzungszeilen nicht mehr angegeben zu werden.

DO und IF können bis zu 20 Stufen tief verschachtelt werden.

IF-DAT(E) Datumsfelder vergleichen, Format ttmmjj

OP F1 OK F2 (BO)

OP IF-DAT oder IF-DATE
F1 erstes Vergleichsdatum
OK Vergleichsoperator (sh. IF)
F2 zweites Vergleichsdatum
BO logische Verknüpfung mit AND und OR

Beispiele:

```
IF-DAT LOEDAT > UDATE AND
IF-DATE DATUM > 311299 OR
   VDAT < 010100 AND
IF BETRAG > 0
```

Zweck:

Vergleich von Datumswerten, 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:

Bei IF-DAT werden intern die Datum-Angaben vor dem Vergleich in das achtstellige ISO-Format JJJJMMTT konvertiert.

IF-DAT kann alphanumerische und numerische Felder oder Konstanten miteinander vergleichen, auch wenn Länge und/oder Typ unterschiedlich sind. 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 der interne Vergleichswert 19970101.

Beachte: IF-DAT prüft die Vergleichsfelder nicht. Der Programmierer ist dafür verantwortlich, dass die Felder im richtigen Format zur Verfügung stehen und gültige Werte enthalten.

Die Operationen IF, IF-DAT und IF-DATI können kombiniert und mit AND und OR logisch verknüpft werden.

Unterstützte Formate für IF-DAT (am Beispiel 31.12.1997):

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
6	311297	311297	19971231
7		0311297	19971231
		1311297	19971231
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: 0 oder 1 für 1900, 2 für 2000.

Die IF-DAT-Operationen arbeiten nach der gleitenden Fenstertechnik mit einem Defaultwert von 30. Das heisst, Jahreszahlen größer als 30 werden dem Jahrhundert 19xx zugeordnet, Jahre kleiner und gleich 30 dem Jahrhundert 20xx. Ein anderes Fenster kann in der Kundenkonfiguration (CPGURSIT,sh. CPG-Installation) vorgegeben werden.

IF-DATI Datumsfelder im ISO-Format vergleichen

OP F1 OK F2 (BO)

OP IF-DATI
F1 erstes Vergleichsdatum
OK Vergleichsoperator (sh. IF)
F2 zweites Vergleichsdatum
BO logische Verknüpfung mit AND und OR

Beispiele:

```
IF-DATI LOEDAT > CPGDAI
IF-DATI ALPHA8 < NUM20
IF-DATI LOEDAT > CPGDAI AND
IF-DATI DATUM > 991231 OR
   VDAT < '000101' AND
IF BETRAG > 0
```

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 entspricht der Operation IF-DAT, aber mit dem Unterschied, dass hier beim Datumswerte im ISO-FORMAT JJMMTT miteinander verglichen werden.

Unterstützte Formate für IF-DATI (am Beispiel 31.12.1997):

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

JLB Linksbündig verschieben, Blanks nach hinten

OP F2

OP JLB oder LEFT-SHIFT muss eingetragen werden
F2 Name eines alphanumerischen Feldes

Beispiel :

JLB FELD
LEFT-SHIFT FELD

Zweck:

Linksbündiges Verschieben eines Feldinhalts.

Beschreibung:

Mit dieser Operation wird der Inhalt eines Alphafelds so verschoben, dass das erste Zeichen ungleich Blank nach der Ausführung in der linkesten Stelle des Feldes steht. Nach rechts wird das Feld mit Blanks aufgefüllt.

Beispiel:	JLB FELD	
Feldinhalt vorher	1. ' 123'	2. ' 1 2 3'
Feldinhalt nachher	'123 '	'1 2 3 '

Achtung: Beispiel 2 erläutert die genaue Arbeitsweise von JLB. Es werden nicht lediglich die Blanks nach rechts sortiert; Zeichenketten bleiben unverändert, auch wenn sie Blanks einschließen. Um in diesem Fall das gleiche Ergebnis wie in Beispiel 1 zu erzielen, müsste vor JLB ein DELC ' ' FELD programmiert werden.

JRB Rechtsbündig verschieben, Blanks nach vorne

OP F2

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.

Beispiel:	JRB FELD	
Feldinhalt vorher	1. '123 '	2. '12 3 '
Feldinhalt nachher	' 123'	' 12 3'

Anwendung: Randausgleich für Textausgaben, ungepackte numerische Daten, usw.

Achtung: 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 JRB ein DELC ' ' FELD programmiert werden.

JRC Rechtsbündig verschieben, bestimmtes Zeichen nach vorne

OP F2 EG

OP JRC oder RIGHT-CHAR muss eingetragen werden
F2 Zeichen, mit dem aufgefüllt wird
EG Name eines alphanumerischen Feldes

Beispiel :

```
JRC  '*'  FELD
RIGHT-CHAR  '-'  FELD2
```

Zweck:

Rechtsbündiges Verschieben eines Feldinhalts.

Beschreibung:

Die Operation JRC arbeitet wie JRB. 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

OP EG

OP JRZ oder RIGHT-ZERO muss eingetragen werden
EG Name eines alphanumerischen Feldes

Beispiel :

JRZ FELD
RIGHT-ZERO KDNR

Zweck:

Rechtsbündiges Verschieben eines Feldinhalts.

Beschreibung:

Die Operation JRZ arbeitet wie JRB. 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

---

identisch mit JLB (s.o.)

LIST Ausgabe extern beschriebener Listen

(F1) OP F2 (DY) (EG) (SV)

F1 Druckername
OP LIST muss eingetragen werden
F2 Name eines QTF-Dokuments
DY Dummywort SECTion
EG Name einer Section im Druckdokument
SV P=List Phase, I=Phase wenn Dokument nicht vorhanden ist.

Beispiele:

```
'L86C' LIST KUNDE
DRID   LIST-DOKUM5 HEADER
DRID   LIST QPG$L  OPCODE
       LIST-DOKUM SECT DETAIL
'QTFS' LIST QPGDL SECTION OPCODE PHASE
```

Zweck:

Drucken einer Liste, die programmextern im QTF (Quick Text Facility) beschrieben ist.

Beschreibung:

In F1 muss (außer bei Batch-Verarbeitung und bei direktem Drucken im CICS) 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 – außer, es wird der Drucker PRDR benutzt – dann wird die Library JOB verwendet. Ein '$'-Zeichen im Dokumentnamen wird durch den Sprachencode (QTF) des Benutzers ersetzt. Damit kann das Programm ohne aufwändige Programmierung für die gültigen Sprachen (z.Zt. D=Deutsch, E=Englisch) benutzt werden.

Zusätzlich kann der Name einer Section angegeben werden. Eine Section ist ein Teil eines Dokuments.

Der aktuelle Linecounter kann nach der LIST-Operation abgefragt werden, wenn im Programm das Feld CPGLCT (3,0 Stellen) definiert ist. CPGLCT kann dann z.B. zur Seitenüberlauf-Steuerung benutzt werden.

LIST-VAR Variable Programmierung des LIST-Befehls

OP EG (SV)

OP LIST-VAR muss eingetragen werden
EG Name eines 32-stelligen Alphafeldes
SV P=List Phase, I=Phase wenn Dokument nicht vorhanden ist.

Zweck:

Der LIST Befehl (wie oben beschrieben) soll variabel programmiert werden. Die notwendigen Daten werden zur Ausführungszeit dem 32-stelligen Feld mit folgendem Aufbau (alle Teilfelder alpha) entnommen:

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 - 26	Automatisches Newpage (S)
Stelle 27 - 32	noch unbenutzt, sollte mit Blanks belegt werden.

Ausführliche Beschreibung siehe QTF-Handbuch.

LOADT Geretteten Bildschirminhalt zurückladen

LOADT-VAR Geretteten Bildschirminhalt variabel zurückladen

OP F2

OP LOADT oder LOADT-VAR muss eingetragen werden
F2 Name einer Temporary Storage Queue

Beispiele:

```
LOADT TS01
LOADT 'HUGO'
LOADT-VAR FELD
```

Zweck:

Laden eines mit SAVET geretteten Bildschirminhalts.

Beschreibung:

Diese Operation schreibt den mit SAVET geretteten Bildschirminhalt wieder zurück auf das Terminal. Es muss der Name der Temporary Storage Queue angegeben werden, auf der mit Operation SAVET die Daten abgestellt wurden. Der Name wird bei LOADT konstant 4-stellig angegeben werden bzw. bei LOADT-VAR in einem Feld, das 4-stellig alphanumerisch definiert ist.

Die Operation LOADT ist nur zusammen mit SAVET sinnvoll, dabei ist es aber nicht erforderlich, dass die beiden Operationen im gleichen Programm verwendet werden.

Bei dialogorientierter Programmierweise hält LOADT das Programm nicht an. Dazu muss ein Bild noch explizit eingelesen werden.

Achtung: Das Zurückladen der Farben ist nur dann gewährleistet, wenn sie per Farbattribut (B,G,P,R,T,W,Y) gesetzt werden.

LOKUP Suchen in einer Feldgruppe

OP F1 OC F2

OP LOKUP muss eingetragen werden
F1 Name der zu durchsuchenden Feldgruppe
OC Operator
F2 Suchargument der zu durchsuchenden Feldgruppe

Beispiele:

```
LOKUP FG(I) = FELD
LOKUP FG(I) <= FELD
LOKUP FG(I) GE FELD
```

Zweck:

Eine Feldgruppe soll auf einen bestimmten Inhalt untersucht werden.

Beschreibung:

LOKUP durchsucht Feldgruppen nach einem bestimmten Inhalt. Die angegebene Feldgruppe wird elementweise mit dem Suchargument verglichen. Sobald die Bedingung erfüllt ist, wird die Operation LOKUP beendet.

Bei der Operation LOKUP muss ein Index angegeben werden, der folgende Funktion erfüllt:

1. Der Vergleich beginnt erst bei dem durch den Inhalt des mit dem Index gekennzeichneten Feldgruppenelementes.

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'
'QPG'
'QSF'
'QTF'
```

Der Befehl
LOKUP FG3(I) = 'QPG'
hat folgende Auswirkungen :

Ist vor dem LOKUP z.B. I=1, dann wird 'QPG' gefunden; damit ist nach dem LOKUP I=3.

Ist vor dem LOKUP z.B. I=4, dann werden nur die Elemente 4 und 5 mit dem Argument 'QPG' verglichen und I=0 gesetzt.

Haben (bei Alphafeldern) das Suchargument und die Feldgruppe unterschiedliche Feldlängen, dann erfolgt der Vergleich in der Länge des kürzeren Operanden.

MAP Mapeingabe übertragen

OP F2 (SV)

OP MAP oder RECEIVE muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV Services: Blank, Clear und Low

Beispiele:

```
MAP KUNDEN.
RECEIVE BRIEF LOW.
```

Zweck:

Aus einer QSF-Map sollen Daten 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 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.

Mit dem Service SV sind folgende Optionen möglich:

Blank Löschen Alphafelder der Maske auf Blank und num. Felder auf 0.

Clear Löschen variabler Attributfelder der Maske auf Blank.

Low Verarbeiten von Kleinbuchstaben. Es wird keine Übersetzung in Großbuchstaben durchgeführt, wenn in der Terminal Control Table des CICS UCTRAN nicht gesetzt ist, bzw. durch die Operation UCTRN OFF im Programm ausgeschaltet wird.

Ein $-Zeichen im Mapnamen wird durch den Sprachencode des Benutzers (QTF) ersetzt. Damit kann das Programm ohne aufwändige Programmierung für die gültigen Sprachen (z.Z. D=Deutsch, E=Englisch) benutzt werden.

Bei MAP, MAP-VAR, MAPD, MAPD-VAR, MAPI und MAPI-VAR wird der Status im internen Feld CPGMRC (Map Return Code, 2 Stellen alpha) gesetzt:

```
'  '      Normale Eingabe
'IC'      Invalid Charater
'NI'      No Input
```

MAP-VAR MAP-Befehl variabel

OP EG (SV)

```
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. Als Operand wird ein mit 16-Byte definiertes Alphafeld eingetragen, das von Stelle 1-8 den Mapnamen enthält. Stelle 12 kann ein 'B' oder 'C' für den Service 'Blank' oder 'Clear' enthalten (sh. Operation MAP) alle anderen Stellen sind reserviert für MAPO.

Der Status wird im Feld CPGMRC gesetzt (sh. MAP Operation).

MAPD Map Dialog

OP F2 (SV)

OP MAPD muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV Service: LOW, CLear, I

Beispiele:

```
MAPD KUNDEN.
MAPD NUMMER LOW
```

Zweck:

Eine QSF-Map soll ausgegeben werden, und anschließend sollen Daten aus dieser Map im Dialog gelesen werden.

Beschreibung:

Dieser Befehl ermöglicht es, alle Bildschirmfelder interaktiv ausserhalb des Programms zu beschreiben, sh. Operation MAP.

Enthält SV das Schlüsselwort 'LOW', so wird vom QSF keine Übersetzung in Großbuchstaben durchgeführt.

Die Schlüsselworte 'CLear' oder 'I' ermöglichen die Abfrage der Taste CLear (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.

Ein $-Zeichen im Mapnamen wird durch den Sprachencode ersetzt.

Der Status wird im Feld CPGMRC gesetzt (sh. MAP Operation).

MAPD-VAR Variabler Map-Dialog

OP EG (SV)

OP MAPD-VAR muss eingetragen werden
EG 16-stelliges Feld mit variablen Informationen
SV Services: LOW, CLear, I

Zweck:

Der MAPD-Befehl (s.o) soll variabel programmiert werden.
Das 16-Byte lange EG-Feld hat folgenden Aufbau:

```
 1 -  8 Mapname
 9 -  9 Erase Y=Yes N=No
10 - 10 Write Control Character:
                                    H = HUPE
                                    K =        MDT + LOCK
                                    L =              LOCK
                                    M =        MDT
                                    N = HUPE + MDT + LOCK
                                    O = HUPE       + LOCK
                                    S = HUPE + MDT
11 - 11 F = Fields Only
12 - 16 reserviert
```

Der Status wird im Feld CPGMRC gesetzt (sh. MAP Operation).

MAPI Eingabe QSF-Map

OP F2 (SV)

OP MAPI muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV Service: LOW, CLear, I und S

Beispiele:

```
MAPI ARTIKEL.
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, sh. Operation MAP.

Enthält SV das Schlüsselwort 'LOW', so wird vom QSF keine Übersetzung in Großbuchstaben durchgeführt.

Die Schlüsselworte 'CLear' oder 'I' ermöglichen die Abfrage der Taste CLear (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.

Ein $ -Zeichen im Mapnamen wird durch den Sprachencode ersetzt.

Der Status wird im Feld CPGMRC gesetzt (sh. MAP Operation).

MAPI-VAR MAPI-Befehl variabel

OP EG (SV)

OP MAPI-VAR muss eingetragen werden
EG 16-stelliges Feld mit variablen Informationen
SV Services: LOW, CLEar, I und S

Zweck:

Der MAPI-Befehl (s.o) soll variabel programmiert werden. Zum Aufbau des 16-stelligen Alphafeldes sh. MAPD-VAR.

Der Status wird im Feld CPGMRC gesetzt (sh. MAP Operation).

MAPO QSF-Map ausgeben

OP F2

OP MAPO oder SEND muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map

Beispiele:

```
MAPO MENUE.
SEND FEHLER.
```

Zweck:

Eine QSF-Map soll ausgegeben werden.

Beschreibung:

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.

Ein $-Zeichen im Mapnamen wird durch den Sprachencode ersetzt.

MAPO-VAR Variable Map-Ausgabe

OP EG

OP MAPO-VAR muss eingetragen werden
EG 16-stelliges Feld mit variablen Informationen

Zweck:

Der MAPO Befehl (s.o.) soll variabel programmiert werden. Zum Aufbau des 16-stelligen Alphafeldes, sh. MAPD-VAR.

MAPP QSF-Map auf einen Drucker ausgeben

F1 OP F2 (SV)

F1 Name des Online-Druckers (als Feld oder als Konstante)
OP MAPP muss eingetragen werden
F2 enthält den bis zu 8-stelligen Namen der Map
SV AFTer, BEFore, S für Vorschub auf den Blattanfang

Beispiele:

```
DRID   MAPP ARTIKEL
'DR15' MAPP POSTEN  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 mindestens ein Zeichen befindet. Sollen also 24 Zeilen gedruckt werden, so muss in jeder Zeile der Map zumindest ein Blank (dargestellt durch ein '#') beschrieben werden.

Folgende Services sind unterstützt:

A - Nach dem MAPP wird ein Vorschub auf Blattanfang durchgeführt.
B – 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

F1 OP EG (SV)

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. Zum Aufbau des 16-stelligen Alphafeldes, sh. MAPD-VAR.

MOVE Rechtsbündig übertragen

```
OP F2 (DY) EG
```

```
OP   MOVE oder MOVE-RIGHT ist einzutragen
F2   Feldname des Ursprungsfeldes
EG   Feldname des Ergebnisfeldes
DY   Dummywort: TO
```

Beispiele:

```
MOVE A B.
MOVE 3,14 TO PI.
MOVE UMS(M) TO UMSATZ.
MOVE FG(1) TO FG(I)
MOVE '*' TO STERN.
MOVE X'FFFFFF' TO ENDTAB.  * Hexadezimale Konstante
```

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.

MOVEA Move Array

```
OP F2 (DY) EG (SV)
```

```
OP   MOVEA oder MOVE-ARRAY muss eingetragen werden
F2   zu übertragender Bereich
DY   Dummywort TO
EG   Feldname des Ergebnisfeldes
SV   Service C (compress) oder E (expand)
```

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 zusammenhängender Bereich. Die Länge der MOVEA–Operation wird durch die Länge des kürzeren Feldes (F2 oder EG) bestimmt.

Mit dem Service C (compress) werden Arrays in eine Zeichenfolge zerlegt, die durch das Zeichen (x'00') getrennt werden. Mit dem Service Expand wird eine Zeichenfolge, deren Elemente durch das Zeichen (x'00') getrennt sind, in die einzelnen Elemente eines Arrays übertragen.

Beispiele :

```
Feldinhalte vorher :  FG25 'ABCDE'   FG34  '****'
                           'FGHIJ'         '****'
                                           '****'

MOVEA  FG25 TO FG34                        'ABCD'
                                           'EFGH'
                                           'IJ**'

MOVE-ARRAY  FG25(2) TO  FG34(2)            '****'
                                           'FGHI'
                                           'J***'

FELD = 'ABC|FG|H|                '
'|' REPLC X'00' FELD
FILL '*' FG34
MOVEA FELD TO FG34 Expand            FG34  'ABC*'
                                           'FG**'
                                           'H***'
```

FILL '*' FELD
MOVEA FG25 TO FELD Compress FELD 'ABCDE|FGHIJ*'**
X'00' REPLC '|' FELD

oder:

```
fg(1) = 'MOVEA     '
fg(2) = 'ist       '
fg(3) = 'richtig   '
fg(4) = 'klasse!   '
movea fg to satz compress
x'00' replace ' ' satz
```

Dann steht linksbündig in SATZ: MOVEA ist richtig klasse!

MOVEL Linksbündig übertragen

```
OP F2 (DY) EG
```

```
OP   MOVEL oder MOVE-LEFT muss eingetragen werden
F2   Feldname des Ursprungsfeldes
EG   Feldname des Ergebnisfeldes
DY   Dummywort: 'TO'
```

Beispiele:

```
MOVEL A TO B.
MOVE-LEFT  A  TO  B
MOVEL 'X' TO TEST.
MOVEL UMS(M) TO UMSATZ.
MOVEL FG(1) TO FG(I).
MOVE-LEFT STERN TO FG.
MOVEL X'0D0C0D' TO ESCAPE.    * Hexadezimale Konstante
```

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.

MOVEN Alphanumerisches in numerisches Feld übertragen

OP F2 (DY) EG (SV)

OP MOVEN muss eingetragen werden
F2 Feldname des alphanumerischen Ursprungsfeldes
EG Feldname des numerischen Ergebnisfeldes
DY Dummywort: 'TO'
SV Service H für erweiterte Feldprüfung

Beispiele:

```
MOVEN A N
MOVEN A TO B
MOVEN UMS(M) TO UMSATZ H
MOVEN FGA(1) TO FGN(I)
MOVEN A TO N H
```

Zweck:

Der Inhalt eines alphamerischen Feldes soll in ein numerisches Feld übertragen werden, und zwar so, dass es der Bildschirmeingabe entspricht. F2 und EG können indiziert oder als komplette Feldgruppen verarbeitet werden.

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 MOVEN entsprechend komprimiert.

Es wird im internen Feld CPGPRC (Program Return Code, 2 Stellen alpha) der Status für die Übertragung gesetzt:

' ' Normale Übertragung.
'BL' Blank: das Alphafeld ist leer.
'IC' Invalid Charater: das Alphafeld hat ungültige Zeichen.
'OF' Overflow: das Alphafeld enthält zuviele Ziffern.

Mit der Servicefunktion 'H' wird eine erweiterte Prüfung ungültiger Zeichen erreicht. Ungültige Zeichen werden auch dann angezeigt, wenn innerhalb der Ziffernfolge Blanks oder Minuszeichen auftreten.

Beispiele: MOVEN ALPHA TO NUM

Das numerische Feld sei 7-stellig mit 2 Dezimalstellen definiert.

ALPHA	NUM	intern	Status	CPGPRC	(mit H)
'123 '	123,00	0012300C		' '	' '
' 123 '	123,00	0012300C		' '	' '
' 1 2 3 '	123,00	0012300C		' '	'IC'
'-123 '	123,00-	0012300D		' '	' '
'12-3 '	123,00-	0012300D		' '	'IC'
' 987,65CR '	987,65-	0098765D		' '	' '
',1 '	0,10	0000010C		' '	' '
'12.345,678'	12345,67	1234567C		'OF'	'OF'
'56,7D '	56,70	0005670C		'IC'	'IC'
'ELF EURO '	0,00	0000000F		'IC'	'IC'
' '	0,00	0000000F		'BL'	'BL'
'12A4567 '	24567,00	2456700C		'OF'	'OF'

MOVEV variable MOVE-Operation

OP F2 (DY) EG (SV)

OP MOVEV muss eingetragen werden
F2 10-stelliges alphanumerisches Feld oder Konstante
EG 10-stelliges alphanumerisches Feld oder Konstante
SV ARRay, LEFt, Numeric, RIGht
DY Dummywort: 'TO' oder 'INTO'

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.
Left für Left. MOVEV arbeitet in diesem Fall wie MOVEL.
Num für numerische Übertragung; **arbeitet wie MOVEL, wenn das Ergebnisfeld alphanumerisch ist.**
Right 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 groß sein. In den ersten 6 Stellen dieser Felder steht immer der Feldname. Bei indizierter Verarbeitung steht also in den Stellen 1-6 der Name der Feldgruppe, in den Stellen 7-10 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)	Num. (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:	Num. (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 im internen Feld CPGPRC (Program Return Code, zwei Stellen alpha) der Status für die Übertragung gesetzt:

' '	Normale Übertragung.
'BL'	Blank: F2 ist leer (bei MOVE Numeric).
'IC'	Invalid Character: in F2 ungültige Zeichen (bei MOVE Numeric).
'IX'	Invalid Index: Bei Feldgruppen ungültiger Index.
'NA'	Not Alphameric: F2 oder EG nicht alpha (bei MOVE Array).
'NA'	Not Alphameric: F2 nicht alpha (bei MOVE Numeric).
'NF'	Not Found: F2 oder EG nicht gefunden.
'NN'	Not Numeric: EG nicht numerisch (bei MOVE Numeric).
'OF'	Overflow: zuviele Ziffern in F2 (bei MOVE Numeric), d.h. bei der Übertragung wurden Ziffern abgeschnitten.

MOVE-ARRAY Feldgruppeninhalt übertragen

identisch mit MOVEA (s.o.)

MOVE-LEFT Feldinhalt linksbündig übertragen

identisch mit MOVEL (s.o.)

MOVE-REST Rest einer Division übertragen

identisch mit MVR (s.u.)

MOVE-RIGHT Feldinhalt rechtsbündig übertragen

identisch mit MOVE (s.o.)

MVR Divisionsrest übertragen

OP EG

OP MVR oder MOVE-REST muss eingetragen werden
EG Feldname des Ergebnisfeldes

Beispiel:

```
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 übertragen werden.

Beispiel:
```
QUOT = 50 / 3
MOVE-REST TO REST
```

Nach dieser Befehlsfolge hat das Feld REST den Inhalt 2.

Der Rest sollte so viele Dezimalstellen haben, wie Quotient und Divisor aus der vorhergehenden Division zusammen. Wenn MVR benutzt wird, darf in der vorhergehenden Division nicht gerundet werden.

OPEN Datei eröffnen

```
OP FN (SV)
```

```
OP   OPEN  muss eingetragen werden
FN   Name der zu eröffnenden Datei
SV   Service (Ein-/Ausgabemode für VSAM-Dateien und Datasets)
```

Beispiele:

```
OPEN CPGWRK.
OPEN KUNDEN.
OPEN DATEI Input
OPEN DATEI Update
OPEN DATASET Output
OPEN DATASET Reuse
```

Zweck:

Eine Datei soll eröffnet werden.

Beschreibung:

Die in FN angegebene Datei wird eröffnet.

Mit dem Service kann bei VSAM-Dateien für die Batchverarbeitung der Ein-/Ausgabemodus spezifiziert werden. Damit kann bei OPEN explizit angegeben werden, ob die Datei für Input, Update, Output oder Reuse (Output) eröffnet werden soll. Der Mode bleibt für die gesamte Batchverarbeitung erhalten.

Bei Datasets wird der Service als Erweiterung des Operationscodes mit übergeben. Z.B. wird 'OI' bei OPEN Input im Feld CPGHIC an ein HL1-Dataset übergeben. Bei einem QPG-Dataset wird z.B. 'OR' bei OPEN Reuse im Feld CPGFRC übergeben.

Der Status nach dem OPEN wird im internen Feld CPGFRC übergeben. Dabei gilt:

```
'  '     Datei wurde erfolgreich geöffnet.
'EF'     Datei ist eine leere Input/Update-VSAM-Datei im Batch
'NF'     Datei wurde nicht (in der FCT) gefunden.
'NO'     Datei konnte nicht eröffnet werden.
```

Bei Datasets muss der Returncode entsprechend vom Dataset bereitgestellt werden.

PERFORM Unterprogramm ausführen

identisch mit EXSR (s.o.)

PROG(RAM) Aufruf QPG-Programm

OP F2 (EG)

OP PROG oder PROGRAM muss eingetragen werden
F2 Programmname
EG Library

Beispiel :

```
PROGRAM OTTO
PROG HUGO TASK
```

Zweck:

Ein QPG-Programm soll als Unterprogramm aufgerufen werden.

Beschreibung:

Das in F2 angegebene QPG-Programm wird als externes Unterprogramm ausgeführt. Dabei werden Felder automatisch übergeben, wenn die Feldnamen und -längen im aufrufenden und aufgerufenen Programm übereinstimmen (siehe Kapitel 3100, Datenaustausch).

In EG wird die Library angegeben, wenn das Unterprogramm sich in einer anderen Library befindet als das aufrufende Programm.

PROG-VAR Aufruf QPG-Programm variabel

OP EG

OP PROG-VAR muss eingetragen werden
EG 32-Byte langes Alphafeld

Beispiel :

```
PROG-VAR HUGO
```

Zweck:

Ein QPG-Programm soll als Unterprogramm aufgerufen werden.

Beschreibung:

Das 32-Byte lange Parameterfeld EG hat folgenden Aufbau:

```
Stelle:  1 -  8 Programmname
         9 - 12 Library    (nur erforderlich, wenn das aufgerufene
                            Programm in anderer Library ist)
        13 - 32 reserviert, sollte mit Blank gefüllt sein.
```

PROT(ECTION) Schutzcode vergeben (siehe Handbuch CPG3-Service, Kapitel Sign On)

OP F2

OP PROT oder PROTECTION muss eingetragen werden
F2 Feld, das den Schutzcode enthält

Beispiel :

```
PROT A
PROTECTION  FELD
```

Zweck:

Ein Programm soll in Verbindung mit CPG3 .. Sign On vor unberechtigtem Zugriff geschützt werden.

Beschreibung:

PROT wird nach neuer Sign-On Logik ausgeführt. Es wird ein 12 - stelliges Alphafeld benötigt, das wie folgt aufgebaut ist:

```
Stelle  1 -  8 : Symbolischer Name des Schutzcodes
             9 : Art der Fehlerbehandlung
                 ' ' -  durch CPG3-Serviceprogramme
                 'R' -  eigene Programmierung nach Abfrage des
                        Return Codes
            10 : Return Code
                 '0' -  Zugriff ist berechtigt
                 '1' -  Non-Terminal-Task, PROT nicht erlaubt
                 '2' -  Benutzer ist nicht angemeldet
                 '4' -  Programm nicht in der Protection-Table
                 '7' -  Fehler im Bereich 'Mandanten'
                 '8' -  Fehler im Bereich 'Sachgebiete'
        11 - 12 : Reserve
```

PURGE Temporary Storage Queue **/ Dataset** löschen

OP F2

OP PURGE muss eingetragen werden
F2 Name zu löschender Temporary Storage Queue **/ eines Datasets**

Beispiel :

```
PURGE STOR
```

Zweck:

Eine Temporary Storage Queue soll gelöscht werden.

RANDOM Datei freigeben

identisch mit RNDOM (s.u.)

READ Lesen sequentiell

(F1) OP FN (SV)

OP READ muss eingetragen werden
F1 Feldname des Schlüssels
FN Dateiname
SV S für Save bei TS

Beispiele:

```
READ CPGWRK
KDNR READ KUNDEN.
READ STORAGE S
1 READ TPTC
READ HQTFC
```

Zweck:

Ein Satz einer Datei soll sequentiell gelesen werden.

Beschreibung:

VSAM (KSDS)

Plattensätze einer indexsequentiell organisierten Datei werden sequentiell gelesen.

In F1 kann man den Feldnamen des Schlüssels angeben. Das Schlüsselfeld kann bei der erstmaligen Ausführung der Instruktion den Schlüssel des Satzes enthalten, mit dem die sequntielle Verarbeitung beginnen soll.

FN 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 Status 'EF' im Feld CPGFRC gesetzt. Bei VSAM-Dateien muss 'EF' nach der READ-Operation abgefragt werden, da bei einem weiteren READ nach EF das Programm mit einer Systemfehlermeldung abbricht.

STORAGE

F1 kann frei bleiben oder einen numerischen Schlüssel enthalten. Im Normalfall wird der Bereich nach dem Lesen freigegeben. Ein 'S' oder 'SAVe' in SV bewirkt bei simulierten TS-Queues, dass der Bereich nach dem Lesen erhalten bleibt.

HL1- und QPG-Datasets

F1 muss frei bleiben.

READ-BACK Datei rückwärts lesen

READB Datei rückwärts lesen

(F1) OP FN

F1 Feldname des Schlüssels kann angegeben werden
OP READB oder READ-BACK muss eingetragen werden
FN Dateiname

Beispiele:

```
KEY READB DATEI.
READ-BACK  KUNDEN
READB HQTFC
```

Zweck:

Ein Satz oder mehrere Sätze z.B. 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, dass der logisch nächste Satz der mit dem nächst kleineren Schlüssel ist.

Bei Datei-Anfang wird der Status 'EF' im internen Feld CPGFRC gesetzt.

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.

READI Segment einer Eingabedatei einlesen

OP FN (DY) EG

OP READI muss eingetragen werden
FN Name der Datei, aus der ein Segment gelesen wird
DY Dummywort: SEGMent
EG Name des Segments, das im Input beschrieben ist

Beispiel:

```
READI AUFTRAG SEGMENT KUNDE
READI AUFTRAG POSITION
```

Zweck:

Aus einer bereits gelesenen Datei soll eine bestimmte Struktur nochmals ins Programm übertragen werden.

Beschreibung:

Übertragen von Segmenten. Insbesondere bei VSAM-Dateien mit verschiedenen Satzarten ist es u.U. von Vorteil, 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. Entsprechend den Eingabebestimmungen für ein Segment werden die Eingabedaten des zuletzt gelesenen Satzes nochmals übertragen.

Achtung: READI ist immer nur nach einer READ- oder READ-BACK-Operation oder nach einem CHAIN-Befehl (wobei im CICS ausserdem der Service 'U' oder 'P' erforderlich ist). Wird diese Vorschrift nicht beachtet, so bricht das Programm bei der CICS-Ausführung ab.

Wird READI im CICS nach CHAIN-U oder CHAIN-P benutzt,ohne dass ein Update erfolgen soll, so ist die Datei nach der letzten READI-Operation mit RNDOM freizugeben.

RECEIVE Transaktionsorientiertes Lesen einer QSF-Map

identisch mit MAP (s.o.)

REPLACE Zeichen ersetzen

REPLC Zeichen ersetzen

(F1) OP F2 EG

F1 Angabe, welche Zeichen ersetzt werden sollen.
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
'012345678' REPLACE '9' SATZ. * Alle Ziffern durch 9 ersetzen
REPLC X'00' F10.              * Blanks durch X'00' ersetzen
REPLC F2 EG.                  * Ersetzen durch Zeichen in F2
'x' REPLC '*'  EG.            * Ersetze 'x' durch '*'
```

Zweck:

Beliebige Zeichen in einem Feld werden durch andere ersetzt.

Beschreibung:

F1 enthält **die** Zeichen, **die** ersetzt werden **sollen**, als alphanumerische Konstante oder als Variable in einem Alphafeld. Ist F1 nicht angegeben, so wird Blank ersetzt.

F2 enthält das Zeichen, das eingesetzt werden soll, als alphanumerische Konstante oder als Variable in einem Alphafeld.

In EG steht der Name des zu verändernden Feldes.

RIGHT Alphafeld rechtsbündig verschieben, vorne Blanks

identisch mit JRB (s.o.)

RIGHT-CHAR Alphafeld rechtsbündig verschieben, vorne Zeichen

identisch mit JRC (s.o.)

RIGHT-ZERO Alphafeld rechtsbündig verschieben, vorne Nullen

identisch mit JRZ (s.o.)

RNDOM Datei freigeben

OP FN

OP RNDOM oder RANDOM muss eingetragen werden
FN Dateiname

Beispiele:

```
RNDOM KUNDEN
RNDOM ARTIKEL
```

Zweck:

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.

Bei FIND-Tabellen erfolgt hiermit ein Zurücksetzen auf den Tabellenanfang. Die Suche bei der nächsten FIND-Operation erfolgt dann wieder bei dem ersten Tabellensatz.

ROLL Feldgruppe verschieben

OP EG

OP ROLL muss eingetragen werden
EG Name einer Feldgruppe (evtl. mit Index)

Beispiel :

```
ROLL  PAGE
ROLL  PAGE(I)
```

Zweck :

Verschieben von Elementen einer Feldgruppe nach vorne

Beschreibung:

Mit ROLL werden innerhalb einer Feldgruppe alle Elemente auf den den nächst niedrigen Index verschoben. Das erste Element der Feldgruppe geht somit verloren; das letzte Element bleibt unverändert. Ist bei der Feldgruppe ein Index angegeben, so beginnt die Verschiebung ab diesem Element.

Beispiel : Inhalt der Feldgruppe FG

	vorher	nach ROLL FG
FG,1	' AAA '	' BBB '
FG,2	' BBB '	' CCC '
FG,3	' CCC '	' CCC '

ROLLB Feldgruppe rückwärts verschieben

ROLL-BACK Feldgruppe rückwärts verschieben

OP EG

OP ROLLB oder ROLL-BACK muss eingetragen werden
EG Name einer Feldgruppe (evtl. mit Index)

Beispiele :

```
ROLLB PAGE
ROLL-BACK FG
ROLL-BACK FG(I)
```

Zweck :

Verschieben von Elementen einer Feldgruppe nach hinten

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. Ist bei der Feldgruppe ein Index angegeben, so beginnt die Verschiebung ab diesem Element.

Beispiel : Inhalt der Feldgruppe FG

	vorher	nach ROLLB FG
FG,1	' AAA '	' AAA '
FG,2	' BBB '	' AAA '
FG,3	' CCC '	' BBB '

SAVET Bildschirminhalt retten

SAVET-VAR Bildschirminhalt variabel retten

OP F2

OP SAVET oder SAVET-VAR muss eingetragen werden
F2 4-stelliger Name einer Temporary Storage Queue

Beispiel :

```
SAVET  TS03
SAVET-VAR FELD
```

Zweck:

Das aktuell angezeigte Bild wird gerettet.

Beschreibung:

Die Operation SAVET rettet den aktuellen Bildschirminhalt in einem Temporary-Storage-Bereich.

F2 enthält in maximal 4 Bytes den Namen einer TS-Queue, in der die Bildschirmseite gerettet wird. (Intern wird dieser Name vorne um die vierstellige Terminal-Id erweitert). Bei SAVET-VAR muss der Name in einem 4-stelligen Alphafeld bereitgestellt werden.

Mit der Operation LOADT kann das gerettete Bild wieder angezeigt werden. Diese Operation wird z.B. benutzt, um aus einem Programm ein Help-Fenster aufzurufen und hinterher den alten Bildschirminhalt wiederherzustellen.

Achtung: Das Zurückladen der Farben ist nur dann gewährleistet, wenn sie per Farbattribut (B,G,P,R,T,W,Y) gesetzt sind.

SCAN Alphafeld nach einer Zeichenfolge durchsuchen

F1 OP F2 EG (SV)

F1 Suchargument (Zeichenfolge)
OP SCAN muss eingetragen werden
F2 Name des Feldes, das durchsucht werden soll
EG Numerisches Feld für Startwert und gefundene Position
SV Service V zum Suchen in var. Länge

Beispiele:

```
FELD  SCAN  FG(I) POS
FG    SCAN  SATZ  INDEX
ARG   SCAN  FELD  INDEX  Var
'*'   SCAN  SATZ  X
```

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 die Position im Ergebnisfeld übergeben.

Als Ergebnisfeld kann ein numerisches Feld mit 0 Dezimalen 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.

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.

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

SDUMP Testhilfe Special Terminaldump

OP (F2)

OP SDUMP, TDUMP oder SCREENDUMP ist einzutragen
F2 4-stelliger Dump-Code zur Identifizierung

Beispiele :

```
SDUMP 1
SCREENDUMP
```

Zweck:

Erzeugen eines Special Dumps auf dem Bildschirm

SELCT Feldauswahl

SELECT Feldauswahl

OP EG (DY F2)

OP SELCT oder SELECT muss eingetragen werden
EG Name des Feldes, aus dem eingelesen wird
DY Dummyworte: SEGMent, TYPE oder SELECT-Type
F2 SEGMENT Name in Input Division

Beispiele:

```
SELECT FELD
SELCT CPGCOM
SELECT CPGCOM TYPE T1
SELECT CPGCOM SELECT-TYPE T2
SELCT CPGSIN
SELCT CPGTCT
SELCT PAGE SEGMENT KOPF
SELCT PAGE SEGM SUMME
SELECT FG(X)
```

Zweck:

In der Input Division wird das Feld wie eine Datei gelesen. Durch Angabe eines Segment-Namens können unterschiedliche Selektionen getroffen werden.

SEND Ausgabe einer QSF-Map auf den Bildschirm

identisch mit MAPO (s.o.)

SETL-LIMIT Zeiger auf einen Satz einer Datei setzen

SETLL Zeiger auf einen Satz einer Datei setzen

F1 OP FN

F1 Feldname des Schlüssels, mit dem positioniert wird
OP SETLL oder SET-LIMIT muss eingetragen werden
FN Dateiname

Beispiele:

```
KEY SETLL CPGWRK
KDNRA SET-LIMIT CPGKDN
QNR SETLL STOR
```

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.

FN 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 nächsthöheren Schlüssel aufgesetzt.

Ist bei der Ausführung einer SETLL-Operation das Ende einer Datei erreicht worden, so wird 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 !

SORT(A) Feldgruppe sortieren

OP F2 (SV)

OP SORTA oder SORT muss eingetragen werden
F2 Name einer Feldgruppe
SV 'Blanks', 'Descending'

Beispiele:

```
SORT  FG1.
SORTA PAGE BLANKS
SORTA FG2  DESCEND
```

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.

Beispiele :

```
SORTA  FG
SORTA  FG DOWN
SORTA  FG BLA
```

	vorher	nach SORTA	nach SORTA D	nach SORTA B
FG(1)	'2'	' '	'2'	'A'
FG(2)	'A'	'A'	'1'	'B'
FG(3)	' '	'B'	'B'	'1'
FG(4)	'B'	'1'	'A'	'2'
FG(5)	'1'	'2'	' '	' '

SQL Verarbeiten einer SQL-Datenbank

SQL-Statement

Mit der Operation SQL kann auf Tabellen in einer SQL-Datenbank zugegriffen werden. Beschreibung sh. Seite 3300.

Folgende SQL-Befehle sind unterstützt:

```
ACCESS   (QPG-Erweiterung von SELECT, INSERT, UPDATE oder DELETE)
ACQUIRE  DBSPACE
ALTER    TABLE oder DBSPACE
CLOSE    CURSOR oder CLOSE INSERT
CLOSE    CURSOR                    (aus Declare, ext. dyn. SQL)
COMMIT
CONNECT
CREATE   INDEX, SYNONYM, TABLE oder VIEW
DECLARE  CURSOR FOR SELECT oder INSERT   (extended dynamic SQL)
DELETE
DELETE   WHERE CURRENT OF CURSOR   (aus Declare, ext. dyn. SQL)
DROP     DBSPACE, INDEX, SYNONYM, TABLE oder VIEW
FETCH
FETCH    CURSOR             (Cursor aus Declare, ext. dyn. SQL)
GRANT
INSERT
LOCK     TABLE oder DBSPACE
OPEN     CURSOR                    (aus Declare, ext. dyn. SQL)
PUT
ROLLBACK
SELECT
SELECT   INTO                              (extended dynamic SQL)
UPDATE   und UPDATE STATISTICS
UPDATE   WHERE CURRENT OF CURSOR   (aus Declare, ext. dyn. SQL)
WHERE    (QPG-Erweiterung von SELECT)
WHEREVAR (QPG-Erweiterung von SELECT)
```

Es ist die Syntax einzuhalten, die im SQL Application Programming Guide beschrieben ist. Fortsetzungszeilen sind mit einem Eintrag in Spalte 72 zu versehen (z.B.'+'). Spalte 71 muss leer bleiben.

Maximal 512 Hostvariablen können in einem SQL-Befehl wie z.B. SELECT oder INSERT verarbeitet werden.

Achtung: Bei Konstanten (z.B. in der IN-Clause eines SELECT Statements) ist die Groß-/Kleinschrift zu beachten.

Bei CONNECT wird geprüft, ob die Datenbank verfügbar ist. Wenn das Feld SQCODE definiert ist, wird bei einem Fehler der Returncode 'SC' in CPGFRC zurückgegeben. In SQCODE ist dann z.B. der SQL-Fehlercode –940 enthalten.

CONNECT TO kann erst ab SQL 3.5 benutzt werden, um die Datenbank für die Verarbeitung während einer LUW auszuwählen.

Beispiele (dynamic SQL)

```
sql connect to :dbname
sql connect :uid identified by :pwd
sql connect :uid identified by :pwd to :dbname

sql acquire public dbspace named qpgtest
sql create table qpgt1                                    +
    ( kdnra   char ( 5) not null unique,                  +
      firma   char (30) not null,                         +
      plz     char ( 5)         ,                         +
      ort     char (20)         ,                         +
      str1    char (25) ) in public.qpgtest
sql create unique index qpgt1i on qpgt1 (kdnra)
sql lock table qpgt1 in exclusive mode

sql cursor insert into test1 (kdnra,firma)                +
    values (:kdnr,:firma )
sql put cursor
sql close insert

sql update test1 set plz = :plz where kdnra = :kdnr

sql delete from test1 where plz is NULL and ort is null

sql select firma,kdnra,plz,ort,str1                       +
    from test1
sql fetch cursor into :firma,:kdnr,:plz:iplz,:ort:iort, +
    :str1:istr1
sql close cursor

sql select count(*) from intadr where rpg in (' ','x')
sql fetch cursor into :count4
sql close cursor

sql update statistics for table qpgt1

sql drop table qpgt1

sql acquire public dbspace named mike
sql alter dbspace public.mike (lock=row)
sql lock dbspace mike in exclusive mode
sql lock dbspace mike in share mode
sql drop dbspace mike

sql create synonym sysdd for system.syscolumns
sql create view dd as select * from sysdd
sql drop synonym sysdd
sql drop view dd

sql commit work release
```

Beispiele mit variablen Where Bedingungen

```
sql access firma, kdnra, plz, ort, str1 from test1
sql wherevar :vwhere

sql access insert into qpgt3 ( i, j )                  +
    select i, j from qpgt2
sql wherevar :vwhere

sql access update qpgt3 set j = 8 - i
sql wherevar :vwhere

sql access delete from qpgt3
sql wherevar :vwhere
```

Tip: Zwischen den einzelnen Parametern sollten Leerzeichen verwendet werden. Das gilt besonders für Parameter, die mit einer Ziffer beginnen. Ausserdem sollte jeder Cursor nach der Verarbeitung geschlossen werden.

Beispiele (extended dynamic SQL)

```
sql declare c01 cursor for                                  +
    insert into sqldba.test1 (firma,kdnra)                  +
    valüs (:firma,:kdnr)
sql open c01
sql put c01
sql close c01

sql declare c02 cursor for                                  +
    select firma                                            +
    from sqldba.test1                                       +
    where kdnra = :kdnr                                     +
    for update of firma, plz
sql open  c02
sql fetch c02 into :firma
sql update sqldba.test1                                     +
    set firma = :firma, plz = :plz                          +
    where current of c02
sql close c02

sql declare c03 cursor for                                  +
    select firma                                            +
    from sqldba.test1                                       +
    where kdnra = :kdnr                                     +
    for update of ort
sql open  c03
sql fetch c03 into :firma
sql delete from sqldba.test1                                +
    where current of c03
sql close c03
```

Achtung: Bei extended dynamischen SQL-Befehlen darf der Cursorname nicht CURSOR oder INSERT sein. Diese Namen sind für dynamische SQL-Befehle reserviert.

Returncodes

Das Ergebnis der SQL-Operation wird im Feld CPGFRC übergeben:

' '	Verarbeitung wurde erfolgreich ausgeführt
'DR'	Beim Hinzufügen wurde 'Duplicate Key' festgestellt.
'DR'	Auf eine SELECT-INTO-Abfrage treffen mehrere Rows zu. Nur die erste Reihe wird in die INTO-Felder gefüllt.
'EF'	Beim sequentiellen Lesen mit Fetch wurde 'End of File' erreicht.
'NF'	Bei Select Into wurden keine Daten gefunden.
'SC'	Ein 'Sonstiger Code' (abnormale Bedingung).

Ist in dem aufrufenden QPG-Programm das Feld SQCODE (7,0) vorhanden, so wird hier der SQL-Fehlercode übergeben. Ist SQCODE nicht definiert, so wird bei Returncode 'SC' das Programm mit einem System-Fehler beendet.

SQRT Quadratwurzel ziehen

SQUARE-ROOT Quadratwurzel ziehen

OP F2 EG (SV)

OP SQRT oder SQARE-ROOT muss eingetragen werden
F2 Name eines numerischen Feldes oder Feldgruppen(element)
EG Name eines numerischen Feldes (für Wurzel)
SV 'H' oder 'ROUnded' zum Runden

Beispiele:

```
SQRT  ZAHL WURZEL
SQARE-ROOT F30 ROUND
```

Zweck:

Die Quadratwurzel einer Zahl soll gezogen werden.

TASK Taskorientierter Programmstart

OP (F2) (EG) (SV)

OP TASK muss eingetragen werden
F2 Name des nächsten Programms
EG Name der nächsten Programm-Library
SV A, B, CL, I, S, T

Beispiele:

```
TASK
TASK HUGO
TASK QTFSORT QPG
TASK EMIL LIB2 CLear
TASK ANNA LIB3 Alternate
```

Zweck:

TASK dient dazu, taskorientiert das nächste QPG-Programm zu starten. In Faktor 2 kann ein neuer Programmname und im Ergebnisfeld eine neue Library eingetragen werden. Wird Faktor 2 nicht angegeben, dann wird das aktuelle Programm aufgerufen. Wird die Library nicht angegeben dann wird die aktuelle Library benutzt.

Er können verschiedene Services angegeben werden:

' '	Standard BS-format. CLear Taste = Ende und Purge TS.
'A'	Alternat. BS-format. CLear Taste = Ende und Purge TS.
'B'	Alternat. BS-format. CLear Taste wird abgefragt.
'CLear'	oder
'I'	Standard BS-format. CLear Taste wird abgefragt.
'S'	Standard BS-format. CLear Taste = Ende mit Save TS.
'T'	Alternat. BS-format. CLear Taste = Ende mit Save TS.

TS ist die zuletzt mit TWA-SAVE gespeicherte private TWA.

Bei TASK erlaubt die Option CLEAR die Abfrage der CLEAR-Taste und hat die gleiche Wirkung wie der Service 'CLear' oder 'I'. Ohne diese Option wird durch die CLEAR-Taste die aktuelle Task im CICS beendet.

TASK-VAR Taskorientierter Programmstart variabel

OP EG (SV)

OP TASK-VAR muss eingetragen werden
EG 32 stell. Feld mit Parametern für nächsten Programmstart.
SV A, B, C, I, S, T

Beispiele:

```
TASK-VAR PARMS
TASK-VAR VTASK CLear
TASK-VAR VTASK Alternate
```

Beschreibung:

TASK-VAR wird wie die Operation Task benutzt,wobei jedoch der Programmname in den Stellen 1-8 und die Library in den Stellen 9-12 des Ergebnisfeldes übergeben werden. Fehlt die Library, dann wird die aktuelle Library benutzt. Der Service entspricht dem der Operation TASK.

TEST-FIELD Feld auf numerische Zeichen prüfen

TESTF Feld auf numerische Zeichen prüfen

OP F2 EG (SV)

OP TESTF oder TEST-FIELD muss eingetragen werden
F2 zu prüfendes Feld
EG Ergebnis von TESTF
SV L, um auch die letzte Stelle zu prüfen (Normalfall)

Beispiele:

```
TESTF FELD STATUS LAST
```

Zweck:

Ein Feld soll auf numerischen Inhalt untersucht werden.

Beschreibung:

Das Feld in F2 wird auf numerischen Inhalt überprüft. Das Ergebnis der Operation wird in EG abgestellt. Dabei bedeutet:

Wenn das Ergebnis alpha ist:

'B' das Feld enthält nur Blanks (nur möglich, wenn F2 alpha ist)
'M' das Feld enthält nur Ziffern und führende Blanks (F2 alpha)
'N' alle Zeichen sind Ziffern.
' ' wird in allen anderen Fällen gesetzt.

Wenn das Ergebnis numerisch ist:

1 das Feld ist numerisch ('B', 'M' oder 'N')
0 das Feld ist nicht numerisch (' ')

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 EBCDI-Codes.

TIME Zeit setzen

OP (EG)

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 numerische Feld mit null Dezimalstellen CPGTIM rechtsbündig übertragen.

TWA-LOAD Private TWA von Temporary Storage einlesen

TWALD Private TWA von Temporary Storage einlesen

OP (F2)

OP TWALD oder TWA-LOAD muss eingetragen werden
F2 Name eines Temporary Storage Bereichs

Beispiel:

```
TWALD
TWA-LOAD
TWALD TS23
```

Zweck:

Eine mit TWA-SAVE auf Temporary Storage gerettete TWA wird wieder eingelesen.

Beschreibung:

In F2 wird ein maximal vierstelliger Name für den TS-Bereich angegeben.

Fehlt F2 dann wird ein interner Name verwendet. Die TWA wird nur gelesen, wenn sie zum Programm passt, d.h. die richtige Länge hat; daher wird TWA-LOAD normalerweise nur in dem Programm ausgeführt, in dem auch die Operation TWA-SAVE verwendet wurde.

Im Feld CPGFRC wird der Status gesetzt, ob die TWA gefunden wurde:

' '	die private TWA wurde geladen.
'EF'	die private TWA wurde nicht gefunden.

Ist CPGEDS in der Data Division definiert, dann werden nur die Felder geladen, die in der privaten TWA vor CPGEDS liegen.

TWA-SAVE Private TWA auf Temporary Storage ausgeben

TWASV TWA auf Temporary Storage retten

OP (F2)

OP TWASV oder TWA-SAVE muss eingetragen werden
F2 Name eines Temporary Storage Bereichs

Beispiel:

```
TWASV
TWA-SAVE
TWA-SAVE TS23
```

Zweck:

Eine TWA wird auf Temporary Storage gerettet.

Beschreibung:

In F2 wird ein maximal vierstelliger Name für den TS-Bereich angegeben. Fehlt F2, dann wird ein interner Name benutzt.

Die mit TWA-SAVE gespeicherte private TWA kann später mit der Operation TWA-LOAD wieder gelesen werden.

Ist CPGEDS in der Data Division definiert, dann werden nur die Felder gespeichert, die in der privaten TWA vor CPGEDS liegen.

UCTRAN Übersetzen in Großbuchstaben

UCTRN Übersetzen in Großbuchstaben

OP F2 (SV)

OP UCTRN muss eingetragen werden
F2 Eintragungen : ON oder OFF
SV Service 'T' zum Übersetzen der CICS-Trans-Ids

Beispiele :

UCTRN ON
UCTRN ON T
UCTRN OFF

Zweck:

Übersetzung ein- oder ausschalten.

Beschreibung:

CICS

Der Eintrag ON in F2 setzt für den Bildschirm, an dem das Programm ausgeführt wird, den Terminal Parameter UCT auf ein. Damit übersetzt der TP-Monitor alle Eingaben von Kleinbuchstaben auf Großbuchstaben.

Der Eintrag OFF schaltet die Übersetzung aus. Somit ist es möglich, transaktionsorientiert auch Texteingaben in Groß-/Kleinschreibung zu erfassen.

Es kann der Service 'T' benutzt werden, wenn nur die CICS-Trans-Ids übersetzt werden sollen.

BATCH

UCTRN ON schaltet die Übersetzung für alle Druckausgaben ein.
UCTRN OFF schaltet die Übersetzung aus.

UPDAT(E) Datensatz in Datei ändern

OP FN (DY EG)

OP UPDAT oder UPDATE muss eingetragen werden
FN Name einer Datei oder Satzbeschreibung
DY Dummywort: SEGMent
EG Segment Name in Output Division

Beispiel :

```
UPDAT CPGWRK
UPDAT AUFTRAG SEGMENT KUNDE
UPDATE AUFTRAG SEGM POSTEN
```

Zweck:

Ändern eines Satzes auf einer Plattendatei.

Beschreibung:

Es kann ein Satz auf einer Datei verändert werden. In der Input Division sind unter FILE FN die zu verändernden Felder anzugeben.

Alternativ kann jedoch das Ausgabeformat außer für HL1–Datasets auch in der Output Division angegeben werden, um z.B. auch Konstanten oder ungepackte Felder auszugeben. Im QPG ist die Output Division nur für Special Datasets unterstützt.

Ist die Datei in der Output Division beschrieben, so erfolgt die Ausgabe nur über die Output Division - ein Eintrag in der Input Division ist dann hierfür nicht erforderlich.

WAIT Warten

OP (F2)

OP WAIT muss eingetragen werden
F2 Numerisches Feld für die Daür des Wartens

Beispiele:

```
WAIT.                  * 1 sec warten
WAIT 5.                * 5 sec warten
WAIT SS.               * warten SS sekunden
```

Zweck:

Warten auf den Timer.

Beschreibung:

Mit WAIT wird ein TIMER gestartet, der nach der in F2 angegebenen Anzahl Sekunden abläuft. Ist F2 nicht angegeben, so wird der Timer auf 1 Sekunde gesetzt. Die Ausführung des Programms wird bis zum Ablauf des Timers suspendiert.

WAIT ermöglicht es,Pausen in zeitaufwändigen Programmabläufen einzubauen, um CICS die Möglichkeit zu bieten, andere Transaktionen zu bedienen. Ein Abbruch mit AICA bei solchen quasi Batch-Anwendungen im CICS kann damit vermieden werden.

WHEN Kennzeichnung einer Bedingung

WHEN-DAT(E) Datumsabfrage Standardformat TTMMJJ in einer Bedingung

WHEN-DATI Datumsabfrage ISO-Format JJMMTT in einer Bedingung

OP F1 OC F2 (SV)

OP	WHEN, WHEN-DAT(E) oder WHEN-DATI muss eingetragen werden
F1	Erstes Vergleichsfeld (indizierbar)
OC	Operater = > < >= <= >< EQ GT LT GE LE NE
F2	Zweites Vergleichsfeld (indizierbar)
SV	Boolesche Verküpfung AND/OR kann hergestellt werden

Die WHEN-Anweisung kennzeichnet innerhalb einer EVALUATE Operation eine alternative Bedingung.

WHEN-DAT und WHEN-DATI erlauben die Abfrage von Datumswerten im Standard- (-DAT) oder im ISO-Format (-DATI). Mit AND/OR können Datumsabfragen miteinander und mit den normalen WHEN-Abfragen verknüpft werden. Siehe auch IF-DAT und IF-DATI.

WHEN OTHER Kennzeichnung einer sonstigen Bedingung

WHEN OTHER

Die WHEN-OTHER-Anweisung kennzeichnet innerhalb einer EVALUATE-Operation die Anweisung, welche durchgeführt wird, wenn alle vorhergehenden WHEN-Bedingungen nicht erfüllt waren.

WRITE Datensatz in Datei neu anlegen

OP FN (DY EG)

OP WRITE muss eingetragen werden
FN Name einer Datei oder Satzbeschreibung
DY Dummywort: SEGMent
EG Segment Name in Output Division

Beispiel :

```
WRITE CPGKDN
WRITE AUFTRAG SEGMENT KUNDE
WRITE AUFTRAG SEGM POSTEN
```

Zweck:

Hinzufügen eines Satzes zu einer Datei.

Beschreibung:

Es kann ein Satz in eine Datei hinzugefügt werden. In der Input Division sind unter FILE FN die zu auszugebenden Felder anzugeben.

Ist der Satz bzw. Schlüssel in der Datei bereits vorhanden, so wird im Feld CPGFRC der Returncode 'DR' übergeben.

Alternativ kann jedoch das Ausgabeformat außer für HL1-Datasets auch in der Output Division angegeben werden um z.B. auch Konstanten oder ungepackte Felder auszugeben. Im QPG ist die Output Division nur für Special Datasets unterstützt.

Ist die Datei in der Output Division beschrieben, so erfolgt die Ausgabe nur über die Output Division, ein Eintrag in der Input Division ist dann hierfür nicht erforderlich.

XFOOT Summe einer Feldgruppe rechnen

OP F2 EG (SV)

OP XFOOT muss eingetragen werden
F2 Name einer numerischen Feldgruppe
EG Name des Summenfeldes
SV 'H' oder 'ROUnded' für Runden

Beispiele:

```
XFOOT FGN SUMME
XFOOT FGN FELD ROUND
```

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. Mit der Servicefunktion wird bestimmt, ob das Ergebnis gerundet wird oder nicht.

Übersicht der Operationen

Opcode	Codierung	Bemerkungen
=	EG = F2 (SV)	Zuweisung SV = rounded
+	EG = F1 + F2 (SV)	SV = rounded
-	EG = F1 - F2 (SV)	SV = rounded
*	EG = F1 * F2 (SV)	SV = rounded
/	EG = F1 / F2 (SV)	SV = rounded
ACCEPT	F1 OP FN	sh. CHAIN
AFOOT	OP FG EG (SV)	SV = ROUnded
AVERAGE	OP FG EG (SV)	sh. AFOOT
BEGSR	F1 OP	
BREAK	OP (SV)	SV = All
CALL	**-- -- -- -- -- -- -- --**	**reserviert für internen Gebrauch** *
CHAIN	F1 OP FN	SV = Check,Prot,Upd RC = CPGFRC
CHANGe	F1 OP F2	
CHECK	OP FN	RC = CPGFRC
CHECK-VAR	OP F2	RC = CPGFRC
CLEAR	OP	
CLOSE	OP FN	RC = CPGFRC
COM-REG	OP F2	sh. COMRG
COMRG	OP F2	
COMPUTE	OP Formel	
CONT	OP	
CONTINUE	OP	sh. CONT
CONVERT	OP F2 (DY) (EG) (SV)	sh. CONVT
CONVT	OP F2 (DY) (EG) (SV)	SV = Char Date Hex Low Sec Time X Year
CPARM	**-- -- -- -- -- -- -- --**	**reserviert für internen Gebrauch** *
DEBUG	OP (F2)	F2 = ON oder OFF
DELC	OP F2 EG (SV)	SV = Array
DELETe	OP F2	
DEQ	**(F1) OP**	*
DISPLAY	F1 OP oder (F1) OP EG	sh. DSPLY
DO	OP (DY) (F1) (F2 DY) (DY EG)	DY = LOOP FROM TIMES WITH
DO UNTIL	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DO UNTIL-DATe	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DO UNTIL-DATI	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DO WHILE	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DO WHILE-DATe	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DO WHILE-DATI	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
DUMP	OP (F2)	F2 = ON, OFF oder Code
DSPLY	F1 OP oder (F1) OP EG	
EDIT	OP (KW) EG	KW = SEGMent oder Type
ELIMinate	OP F2 EG	
ELSE	OP	
END	OP (F2)	
END-EVALUATE	OP	sh. ENDEV
ENDDO	OP (F2)	
ENDEV	OP	
ENDIF	OP	
ENDPR	OP	
ENDSR	OP	
ENQ	**OP (F2)**	*
EVALUATE	OP	

Opcode	Codierung	Bemerkungen
EXHM	OP F2 (EG) (SV)	EG = Datenkanal, SV = I,T
EXIT-TRANS	OP F2	sh. EXITT
EXITD	OP EG (SV)	SV = T
EXITI	OP F2	
EXITT	OP F2	
EXPR	**-- -- -- -- -- -- -- --**	**reserviert für internen Gebrauch** *
EXSR	OP F2	
FILL	OP F2 (DY) EG	DY = INTO
FIND	(F1) OP FN	
GETHS	OP	
HTMLI	OP	
HTMLO	OP (F2)	
IF	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
IF-DATe	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
IF-DATI	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
JLB	OP EG	
JRB	OP EG	
JRC	OP F2 EG	
JRZ	OP EG	
LEFT-SHIFT	OP EG	sh. JLB
LIST	(F1) OP F2 (DY EG) (SV)	SV = I,P
LIST-VAR	OP EG (SV)	SV = I,P
LOADT	OP F2	
LOADT-VAR	OP F2	
LOKUP	OP F1 OC F2	OC = > < >= <= >< EQ GT LT GE LE NE
MAP	OP F2 (SV)	SV = Low
MAP-VAR	OP EG (SV)	SV = Low
MAPD	OP F2 (SV)	SV = Low,Clear,I,S
MAPD-VAR	OP EG (SV)	SV = Low,Clear,I,S
MAPI	OP F2 (SV)	SV = Low,Clear,I,S
MAPI-VAR	OP EG (SV)	SV = Low,Clear,I,S
MAPO	OP F2	
MAPO-VAR	OP EG	
MAPP	F1 OP F2 (SV)	SV = Aft,Bef,S
MAPP-VAR	F1 OP EG (SV)	SV = Aft,Bef,S
MOVE	OP F2 (DY) EG	DY = INTO TO
MOVE-ARRAY	OP F2 (DY) EG	sh. MOVEA
MOVE-LEFT	OP F2 (DY) EG	sh. MOVEL
MOVE-REST	OP EG	sh. MVR
MOVE-RIGHT	OP F2 (DY) EG	sh. MOVE
MOVEA	OP F2 (DY) EG	DY = INTO TO
MOVEL	OP F2 (DY) EG	DY = INTO TO
MOVEN	OP F2 (DY) EG (SV)	DY = INTO TO, SV = A,L,N,R
MOVEV	OP F2 (DY) EG	DY = INTO TO
MVR	OP EG	
OPEN	OP FN (SV)	SV = Inp, Upd, Out, Reuse
PERFORM	OP F2	sh. EXSR
PROGram	OP F2 (EG)	EG = Library
PROG-VAR	OP EG	
PROTection	OP F2	
PURGE	OP F2	
RANDOM	OP FN	sh. RNDOM

Opcode	Codierung	Bemerkungen
READ	(F1) OP FN (SV)	RC = CPGFRC, SV = S
READ-BACK	(F1) OP FN	sh. READB
READB	(F1) OP FN	RC = CPGFRC
READI	OP FN (KW) EG	KW = SEGMent, EG = Input Satzart
RECEIVE	OP F2 (SV)	sh. MAP
REPLACE	OP F2 EG	sh. REPLC
REPLC	OP F2 EG	
ROLLB	OP EG	
RIGHT	OP EG	sh. JRB
RIGHT-CHAR	OP F2 EG	sh. JRC
RIGHT-ZERO	OP EG	sh. JRZ
RNDOM	OP FN	
ROLL	OP EG	
ROLL-BACK	OP EG	sh. ROLLB
ROLLB	OP EG	
SAVET	OP F2	
SAVET-VAR	OP F2	
SCAN	F1 OP F2 EG	
SCREENDUMP	OP (F2)	sh. SDUMP
SDUMP	OP (F2)	
SELCT	OP (KW) EG	KW = SEGMent oder Type
SELECT	OP (KW) EG	sh. SELCT
SEND	OP F2	sh. MAPO
SET-LIMIT	F1 OP FN	sh. SETLL
SETLL	F1 OP FN	RC = CPGFRC
SORTa	OP F2 (SV)	SV = Bla, Desc
SQL	OP Statement	Fortsetzungszeilen in Sp. 72 '+'
SQRT	OP F2 EG (SV)	SV = Round
SQUARE-ROOT	OP F2 EG (SV)	sh. SQRT
TASK	OP (F2) (EG) (SV)	SV = A, B, I, S, T
TASK-VAR	OP EG	SV = A, B, I, S, T
TESTF	OP F2 EG (SV)	SV = L
TEST-FIELD	OP F2 EG (SV)	sh. TESTF
TIME	OP (EG)	
TWA-LOAD	OP (F2)	sh. TWALD
TWA-SAVE	OP (F2)	sh. TWASV
TWALD	OP (F2)	
TWASV	OP (F2)	
UCTRAN	OP F2 (SV)	sh. UCTRN
UCTRN	OP F2	F2 = ON, OFF SV = T
UPDATe	OP FN (KW) EG	RC = CPGFRC, KW = SEGMent, EG = Satzart
WAIT	OP (F2)	
WHEN	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
WHEN-DATe	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
WHEN-DATI	OP F1 OC F2 (SV)	OC = > < >= <= >< EQ GT LT GE LE NE
WRITE	OP FN (KW) EG	RC = CPGFRC, KW = SEGMent, EG = Satzart
XFOOT	OP F2 (DY) EG (SV)	DY = TO, SV = Round

OP = Operation OC = Abfrage SV = Service () wahlweise
F1 = Faktor 1 F2 = Faktor 2 EG = Ergebnis
DY = Dummywort FN = Filename RC = Returncode

Daten / private Work Area

Jedes QPG-Programm erhält beim Aufruf eine eigene private Work Area (PWA), in der die Datenfelder, -feldgruppen und intern benutzte Felder angelegt werden. Hierbei werden die Felder initialisiert, d.h. Alphafelder werden mit Blanks und numerische Felder mit 0 (gepackt) formatiert. Die private Workarea kann temporär (nur für die Dauer der Programmausführung) oder permanent (für die gesamte Dauer der Transaktion) angelegt werden.

Jedes Feld ist durch seinen Feldnamen, Positon, Feldlänge und Anzahl Elemente (bei Feldgruppen) eindeutig identifiziert. Jedes Feld erhält normalerweise seinen eigenen Speicherplatz zugewiesen, aber durch Überlagerung z.B. mit ORG in der Data Division können sich mehrere Felder einen gemeinsamen Speicherplatz teilen.

Die maximale Größe der PWA ist 32.767 Bytes inklusive Praefix und interne Felder. Feldgruppen können maximal 10.000 Elemente enthalten, sofern der Platz in der PWA hierzu ausreicht.

Datenformate

Alphafelder werden im Characterformat und numerische Felder im gepackten Format gespeichert.

Feldlänge

Maximal 256 Bytes bei Alphafeldern und maximal 15 Ziffern mit 0 bis 9 Dezimalstellen bei numerischen Feldern.

Die Feldlänge wird bei der Generierung des Object-Codes geprüft.

Datenaustausch

Die Daten sind die Schnittstelle eines Programms nach 'außen'. Hierüber werden außerdem Informationen über Art der Verarbeitung und Returncodes ausgetauscht. Ein Programmobjekt mit Daten und Funktionen ist vergleichbar einem Atom mit Schale und Kern. Ein Programmobjekt ist die kleinste Einheit einer QPG Software Application Library.

Der Austausch von Informationen erfolgt über den Feldnamen.

Jedes Feld muss eindeutig definiert sein, d. h. sowohl im aufrufenden als auch im aufgerufenen Programm müssen die Felder die gleiche Länge und die gleiche Anzahl Dezimalstellen enthalten. Felder, die unterschiedlich definiert sind, werden nicht ausgetauscht. Ebenso werden CPG-interne Felder nicht ausgetauscht.

Bei Feldgruppen, deren Elemente gleich lang definiert sind, die aber eine unterschiedliche Anzahl von Elementen enthalten, werden nur so viele Elemente übertragen, wie in der kleineren Feldgruppe vorhanden sind.

Gleiche Felder, die in beiden Programmen vorkommen, werden bei Aufruf automatisch an das Unterprogramm übergeben und beim Return wieder zurückgegeben. Die Position eines Feldes spielt bei der Datenübergabe keine Rolle.

Private Felder

Es kann sinnvoll oder notwendig sein, bestimmte Felder vom Datenaustausch auszuschließen. Vor den privaten Feldern ist hierzu das Feld CPGEDS (1 Byte) anzulegen. Felder, die in der PWA hinter CPGEDS liegen, werden nicht übergeben.

CPGEDS wird ebenfalls bei den Operationen TWA-SAVE und TWA-LOAD verwendet.

Options HTML

Felder, die aus einem Browser bei einer Intra-/Internetanwendung übergeben werden, erfordern HTML in den Options. Wird Options HTML in einem untergeordneten QPG-Programm, das mit PROGram aufgerufen wurde, eingetragen, dann werden zuerst die Felder aus dem Browser übergeben, bevor die Felder vom rufenden Programm übergeben werden. Felder können damit in der übergeordneten Stufe verändert werden und stehen damit dem untergeordneten Programm geändert zur Verfügung.

Beispiel zum Datenaustausch:

```
*-------------------------------------------------------------------------
*              PROGRAMM A
*-------------------------------------------------------------------------
DATA DIVISION
         SATZ               80
         WERT                7 2
         INFO               20
         FG        24  *    80
         INDEX               3 0
PROCEDURE DIVISION
         FILL 'A' TO SATZ
         FILL 'A' TO FG
         WERT = 47,11
         INFO = 'DAS IST EIN TEST'
         INDEX = 999
         PROG B
         DEBUG


*-------------------------------------------------------------------------
*              PROGRAMM B
*-------------------------------------------------------------------------
DATA DIVISION
         INFO               79
         WERT                7 2
         FG        12  *    80
         SATZ               80
         CPGEDS              1
         INDEX               3 0
PROCEDURE DIVISION
         DEBUG
         FILL 'B' TO SATZ
         FILL 'B' TO FG
         WERT = 08,15
         INFO = 'HIER IST PROGRAMM B'
         INDEX = 123
```

Programm A initialisiert die Felder SATZ, WERT, INFO, FG und INDEX und ruft danach Programm B auf. Mit DEBUG im Programm B lässt sich feststellen,dass die Felder SATZ und WERT übergeben werden. Von der Feldgruppe FG werden 12 Elemente übergeben. Diese Felder haben in beiden Programmen die gleiche Länge. Die Reihenfolge der Felder spielt keine Rolle. Das Feld INFO wird nicht übergeben, weil es eine andere Länge hat. Das Feld Index wird nicht übergeben, weil es als privates Feld hinter dem internen Feld CPGEDS gespeichert ist.

Die Felder SATZ, WERT, INFO und INDEX und die Feldgruppe FG werden im Programm B neu gefüllt. Nachdem alle Anweisungen im Programm B ausgeführt wurden, kehrt die Kontrolle wieder zu Programm A zurück. Hier kann jetzt mit DEBUG festgestellt werden, dass die Felder SATZ und WERT, die beim Aufruf an Programm B übergeben wurden, bei Rückkehr zu Programm A geändert wieder zurückgegeben wurden. Die anderen Felder INFO und INDEX haben noch die Werte, die vor Aufruf von Programm B gefüllt wurden.

Interne Felder

Feld		Beschreibung	Länge,Dezimalen
CPGCOM	*	Common Area (CICS)	max. 32.750
CPGDAI		ISO Datum, JJJJMMTT	8,0
CPGDAT		Datum TTMMJJJJ	8,0
CPGEDS		Ende Data Segment, private Felder	var.
CPGEOJ	*	Batch Returncode für JCL	4,0
CPGFRC	*	File Return Code	2
CPGHPN		Program Name	8
CPGHTM		HTML (Hyper Text Markup Lang.) Map	16
CPGIOA	*	Input/Output Area für QPG-Datasets	var.
CPGKEY	*	interner Key für QPG-Datasets	64
CPGMCU	*	Map Cursor Field	6
CPGMCI	*	Map Cursor Index	3,0
CPGMFN	*	Map Field Name	6
CPGMFI	*	Map Field Index	3,0
CPGMLC	*	Map Location of Cursor	4
CPGMPF		Map Program Function DE,P1,CL ...	2
CPGMPL	*	Map Language	1
CPGMRC		Map oder MOVE Return Code IC, NI	2
CPGPRL		Program Library	4
CPGSIN		System-Informationen	155
CPGTCA	*	Task Control Area	var.
CPGTCT	*	TCT User Area	max. 245
CPGTID		Terminal-Id	4
CPGTIM		Uhrzeit hhmmss	6,0
CPGTSN	*	Variabler TS-Name	8
CPGVRL	*	Variable Satzlänge	5,0
PARMS		Startparameter von QPGUTIL	256
PRDATA		Übergabeparameter an QPGUTIL	max. 40 * 1
SQCODE		SQL Returncode	7,0
SQLCAF		SQLCA Felder	120
SQLISL		SQL Isolation Level	1
UDATE		System-Datum	8
UDATEC		System-Datum, Jahrhundert	2
UDATEI		ISO-Datum, JJJJMMTT	8
UDAY		System-Tag	2,0
UMONTH		System-Monat	2,0
UTIME		System-Zeit	8
UYEAR		System-Jahr	2,0

Die internen Felder werden automatisch definiert, wenn sie benutzt werden. Sie können abgefragt oder übertragen werden. Nur die mit * gekennzeichneten Felder dürfen im Programm modifiziert werden. Die Map-Felder werden definiert, wenn eine MAP-Operation benutzt wird.

CPGCOM

Mit den Operationen EDIT CPGCOM und SELECT CPGCOM kann eine Common Area im CICS mit einer Länge von maximal 32.750 Bytes verarbeitet werden. Der Standardwert bei der Ausgabe ist 4.080. Wird eine höhere Ausgabeposition angegeben, dann wird die maximale Ausgabeposition verwendet. Wird bei der Ausgabe der Parameter VAR angegeben, dann wird als Ausgabelänge der Wert des internen Feldes CPGVRL verwendet.

Ist die Ausgabeposition höher als der Inhalt von CPGVRL, dann wird das Programm bei der Ausführung mit einer Fehlermeldung abgebrochen. Sobald eine Common Area vorhanden ist, kann die Länge in der laufenden Task nicht mehr geändert werden.

Wenn das interne Feld CPGVRL mit 0 initialisiert ist, dann wird die Länge der Common Area in CPGVRL nach Eingabe mit SELCT CPGCOM zurückgegeben.

CPGFRC

Dieses Feld steuert alle Ein- und Ausgabeoperationen. Bei QPG- und HL1-Datasets wird hier bei Aufruf ein Operationscode übergeben. Diese sind:

'C '	Close
'D '	Rndom
'G '	Chain
'GC'	Chain Check
'GP'	Chain P (Check + Update)
'GU'	Chain Update
'I '	Readi
'L '	Delete
'N '	Write
'O '	Open
'OI'	Open Input
'OU'	Open Update
'OO'	Open Output
'OR'	Open Reuse/Replace
'P '	**Purge**
'R '	Read
'U '	Update
'Z '	Check
'ZF'	Check Variable

Nach der Operation wird hier ein Returncode gesetzt. Diese sind:

' '	Alles ok
'DR'	Duplicate Record nach Write
'DR'	Duplicate Record bei SQL INSERT und SQCODE -803.
'DR'	Duplicate Record bei SQL SELECT INTO und SQCODE -810.
'EF'	End of File nach Read oder Readb
'EF'	Empty File nach Open
'EF'	End of File bei SQL FETCH oder SELECT INTO und SQCODE 100.
'NF'	Not Found nach Chain
'NF'	Not Found nach Open
'SC'	SQL Check, bei SQL wenn SQCODE nicht 0, 100, -803 oder –810 ist.

Datasets können hier zusätzlich noch eigene Returncodes setzen.

CPGMPF

Die bei einer Bildschirmeingabe benutzte Funktionstaste kann im Feld CPGMPF abgefragt werden. CPGMPF (2 Stellen, alpha) kann dabei folgende Werte enthalten:

'A1' - 'A3'	Programmabruftasten PA1, PA2 und PA3
'CL'	Löschtaste
'DE'	Datenfreigabe
'P1' - 'P9'	Funktionstaste PF1 - PF9
'PA' - 'PC'	Funktionstaste PF10 - PF12
'Q1' - 'Q9'	Funktionstaste PF13 - PF21
'QA' - 'QC'	Funktionstaste PF22 - PF24
'SP'	Lichtstift oder Positionsauswahl

CPGTCT

Mit den Operationen EDIT und SELCT für das interne Feld CPGTCT kann die TCT User Area benutzt werden. Bei EDIT wird bei Ausführung die verfügbare Länge (maximal 245 Stellen) der TCT User-Area geprüft (nur bei Command Level möglich). Ist keine TCT User Area installiert, ist die Länge zu klein oder wird EDIT CPGTCT in einer Macro-Level-Umgebung aufgerufen, dann wird das Programm mit einer Fehlermeldung abgebrochen.

SQL-Datenbank

Mit der Operation SQL kann jetzt auf Tabellen in einer SQL-Datenbank (z.Zt. nur VSE) zugegriffen werden. SQL-Aufrufe können nur in der CPG-ESA-Commandlevel-Umgebung ablaufen. Folgende SQL-Befehle sind unterstützt, falls der betreffende SQL-User die entsprechende Berechtigung besitzt:

ACQUIRE	DBSPACE zum Anlegen eines Datenbank-Speicherbereichs	*DBA
ALTER	DBSPACE zum Ändern eines Datenbank-Speicherbereichs	*DBA
ALTER	TABLE zum Anlegen neuer Felder	*DBA
CLOSE	CURSOR zum Schließen eines Cursors bei SELECT und FETCH	
CLOSE	INSERT zum Schließen bei CURSOR INSERT und PUT.	
CLOSE	CURSOR um Ein-/Ausgabe-Cursor zu schließen. (ext. dyn. SQL)	
COMMIT	Freigabe der Logical Unit of Work (auch mit RELEASE Option)	
CONNECT	Anmelden ans SQL mit User-Id und Passwort (und DB-Name ab SQL 3.5)	
CREATE	INDEX zum Anlegen eines Index	*DBA
CREATE	SYNONYM zum Anlegen eines Alias-Namen	*DBA
CREATE	TABLE zum Anlegen einer Tabelle	*DBA
CREATE	VIEW zum Anlegen einer Datensicht	*DBA
DECLARE	CURSOR FOR INSERT definiert Cursor zur Ausgabe (ext. dyn. SQL)	
DECLARE	CURSOR FOR SELECT definiert Cursor zur Eingabe (ext. dyn. SQL)	
DECLARE	CURSOR FOR SELECT ... FOR UPDATE OF Ändern/Löschen (ext.d. SQL)	
DELETE	Löschen in Tabellen	
DELETE	WHERE CURRENT OF CURSOR zum Löschen von Daten (ext. dyn. SQL)	
DROP	DBSPACE zum Löschen eines Datenbank-Speicherbereichs	*DBA
DROP	INDEX zum Löschen eines Index	*DBA
DROP	SYNONYM zum Löschen eines Alias-Namen	*DBA
DROP	TABLE zum Löschen einer Tabelle	*DBA
DROP	VIEW zum Löschen einer Datensicht	*DBA
FETCH	Übertragen der SQL Felder in die Host-Variablen	
FETCH	CURSOR s.o. jedoch mit Cursor aus DECLARE (ext. dyn. SQL)	
GRANT	Weitergabe von Zugriffsberechtigungen	*DBA
INSERT	Einfügen von Zeilen in eine Tabelle (auch mit Cursor)	
LOCK	DBSPACE zur Zugriffskontrolle eines Datenbank Speicherbereichs	*DBA
LOCK	TABLE um die Zugriffskontrolle festzulegen	
OPEN	CURSOR um Cursor zur Ein- oder Ausgabe zu öffnen (ext. dyn. SQL)	
PUT	Einfügen von Zeilen bei CURSOR INSERT	
ROLLBACK	Rücksetzen der Logical Unit of Work (auch mit RELEASE-Option)	
SELECT	Abfrage von Tabellen	
SELECT	INTO um einen Satz aus Tabellen zu lesen (ext. dyn. SQL)	
UPDATE	Ändern von Tabellen.	
UPDATE	STATISTICS zum Fortschreiben der Statistik bei Tabellen	
UPDATE	WHERE CURRENT OF CURSOR zum Ändern von Daten (ext. dyn. SQL)	

Die Befehle mit *DBA setzen eine spezielle Autorisierung voraus !

Folgende Pseudo-Operationen (nur im QPG) erlauben variable Abfragen von SQL-Tabellen:

ACCESS	Definition der SQL-Variablen für SELECT, INSERT, UPDATE oder DELETE
WHERE	WHERE-Bedingung für SELECT
WHEREVAR	Variable WHERE-Bedingung für SELECT, INSERT, UPDATE oder DELETE. Die Variable kann auch eine Feldgruppe sein. Bei Feldgruppen ist zwischen den Elementen mindestens 1 Blank erforderlich.

Pro SQL-Statement können bis zu 512 Host-Variablen verarbeitet werden. Z.Zt. können in einer LUW bei dynamischen Zugriffen maximal 10 Cursor zum Lesen oder für CURSOR INSERT eröffnet sein, dabei kann je QPG-Programm in der LUW immer nur ein Cursor zum Lesen und ein Cursor zum Schreiben gleichzeitig eröffnet sein. Nach dem Close eines Cursors kann aber im gleichen Programm ein Cursor z.B. zum Lesen erneut eröffnet werden. Bei extended dynamischen Zugriffen gilt diese Einschränkung nicht.

Das Ergebnis der SQL-Operation wird im Feld CPGFRC übergeben:

' '	Verarbeitung wurde erfolgreich ausgeführt
'DR'	Beim Hinzufügen wurde Fehler 'Duplicate Key' festgestellt.
'DR'	Auf eine SELECT INTO Abfrage treffen mehrere Rows zu. Dabei wird nur die erste Reihe in die INTO Felder gefüllt.
'EF'	Beim seq. lesen mit Fetch wurde das 'End of File' erreicht.
'NF'	Bei Select Into wurden keine Daten gefunden.
'SC'	Ein 'Sonstiger Code' wurde festgestellt (abnormale Bedingung).

Ist in dem aufrufenden QPG-Programm das Feld SQCODE (7,0) vorhanden, so wird hier der SQL-Fehlercode übergeben. Ist SQCODE nicht definiert, so wird bei Returncode 'SC' das Programm mit einem System-Fehler beendet.

Achtung: Wegen der Mächtigkeit der SQL-Befehle ist dringend zu empfehlen, die QPG-Programme mit SQL-Commands in einer Testumgebung mit einer Testdatenbank zu entwickeln und auszutesten, bevor diese Programme in die Produktionsumgebung übernommen werden.

SQLCA SQL Control Area

Nach einem SQL-Befehl kann die SQL Control Area im QPG-Programm mit Select SQLCAF analysiert werden. Alle Variablen der SQLCA stehen zur Verfügung.

SQLCAF wird automatisch mit 120 Stellen definiert, wenn es im Programm benutzt wird.

Beispiel:

```
input division
         field sqlcaf define
                        bin    1    2 0 sqelen.  * Länge der Feldes SQERRM
                               3   72   sqerrm.  * Error Message(s)
                              73   80   sqerrp.  * internes sql modul
                        bin   81   84 0 sqerd1.  * rds error code
                        bin   85   88 0 sqerd2.  * database system return code
                        bin   89   92 0 sqerd3.  * anzahl modifizierte zeilen
                        bin   93   96 0 sqerd4.  * ungefährer cost faktor
                        bin   97  100 0 sqerd5.  * anz abhängige zeilen bei del
                        bin  101  104 0 sqerd6.  * reserviert
                             105  115   sqwarn.  * warning anzeiger
                             116  120   sqstat.  * database returncode
procedure division
         ...
         sql select ...
         if sqcode >< 0
            select sqlcaf
         endif
         ...
```

Die Bedeutung der SQLCA-Felder ist im Handbuch 'SQL Reference for IBM VM System and VSE' ausführlich beschrieben. Eine Kurzbeschreibung bietet das QSAT-Handbuch.

Datentypen

CHAR	Hostvariablen sind Alphafelder
DATE	Hostvariablen sind Alphafelder, die mindestens 10 Bytes lang sind. Angabe z.B.: 'dd.mm.yyyy' im IBM European Standard Format.
DECIMAL	Hostvariablen sind numerische Felder
INTeger	Hostvariablen sind numerische (ohne Dezimalstellen)
NUMERIC	Hostvariablen sind numerische Felder
SMALLINT	Hostvariablen sind numerische Felder (ohne Dezimalstellen)
TIME	Hostvariablen sind Alphafelder, die mindestens 5 bzw. 8 Bytes lang sind Angabe z.B.: 'hh.mm.ss' im IBM European Standard Format.
TIMESTAMP	Hostvariablen sind Alphafelder, die mindestens 19 bzw. 26 Bytes lang sind.
VARCHAR	Hostvariablen sind Alphafelder oder -Feldgruppen. Bei Feldgruppen ist die Länge gleich der Gesamtlänge der Feldgruppe.

Data Dictionary

Dateien müssen im Data Dictionary beschrieben sein. In der Files-Division wird auf die Definitionen mit dem Namen und gegebenenfalls TYPE (Satzart) zugegriffen. Der Mode-Parameter (INP, UPD, OUT) kann jedoch überschrieben werden.

Strukturen werden mit DEFINE in der Data-Division oder mit DD in der Input und Output-Division beschrieben. Der Vorteil dieser Technik besteht darin, dass die Daten und Strukturen zentral im DD gepflegt werden können und dass jeweils der aktuelle Stand in den Programmen verfügbar ist, ohne dass explizit alle Programme einzeln angefasst werden müssen.

Mit dem Data Dictionary können Felder zentral definiert werden. In QPG-Programmen können diese Felder benutzt werden, ohne dass hierzu eine Definition in der DATA oder INPUT DIVISION erforderlich ist. Diese Felder werden automatisch in der im DD festgelegten Länge angelegt, wenn sie im Programm benutzt werden und hier nicht explizit definiert sind. Um das zentrale Data Dictionary nutzen zu können, muss beim Anlegen einer QPG-Programm-Library das Data Dictionary mit '*' aktiviert werden.

Es sind ebenfalls lange Feldnamen mit dem Data Dictionary unterstützt. Sobald ein Feldname im Programm verwendet wird, der länger ist als 6 Stellen, wird er durch den im DD angegebenen Kurznamen ersetzt. Wird der Langname im DD nicht gefunden, so werden die ersten 6 Stellen des Feldnamens als Kurzname verwendet. Langnamen können nicht als Indexnamen benutzt werden.

Programmtest

QPG-Programme können interaktiv getestet werden. Hierzu dient die Funktion DEBUG, die in den Options des Programms, als Operation in der Procedure Division oder extern mit der Service-Transaktion QPG ein- oder ausgeschaltet werden kann.

Eine andere Testmöglichkeit besteht darin, eine Programmlibrary an Testbildschirmen durch eine Testlibrary zu ersetzen. Jeder Zugriff, der an einem Mastertermi-nal zu einer Produktionslibrary erfolgt, wird automatisch zu einer Testlibrary umgeleitet. Diese TEST-Funktion wird mit der Transaktion QPG und dem Befehl TEST aktiviert.

Debug

DEBUG ist eine komfortable Testhilfe, die es ermöglicht, einen Programmablauf interaktiv zu verfolgen und Fehler schnell und einfach zu lokalisieren und zu beheben.

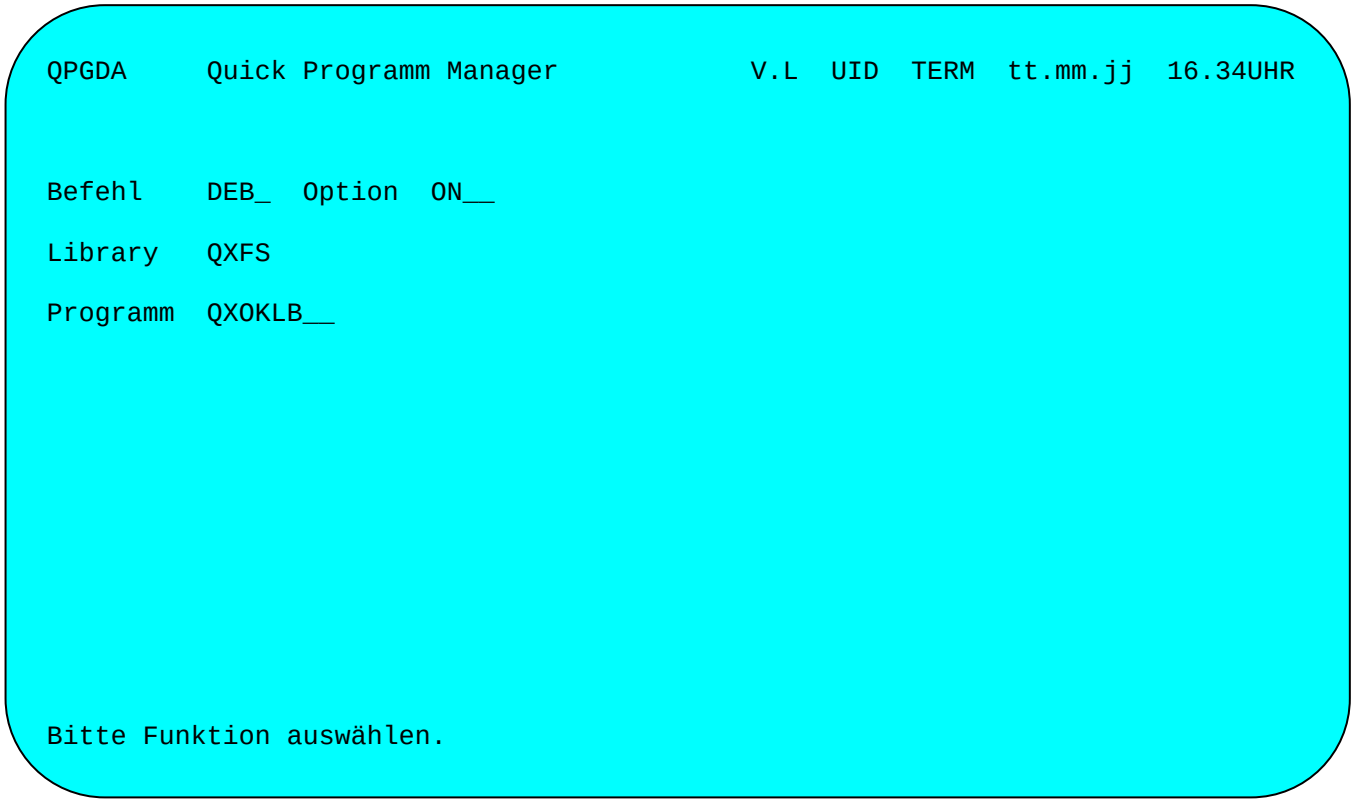
Die Testhilfe wird aktiviert durch die Angabe

```
OPTIONS DEBug
```

oder in der Procedure Division durch

```
-C
     DEBUG                      * nur 1 Statement oder
     :
     DEBUG ON
     :                          * zu testender Programmteil
     DEBUG OFF
```

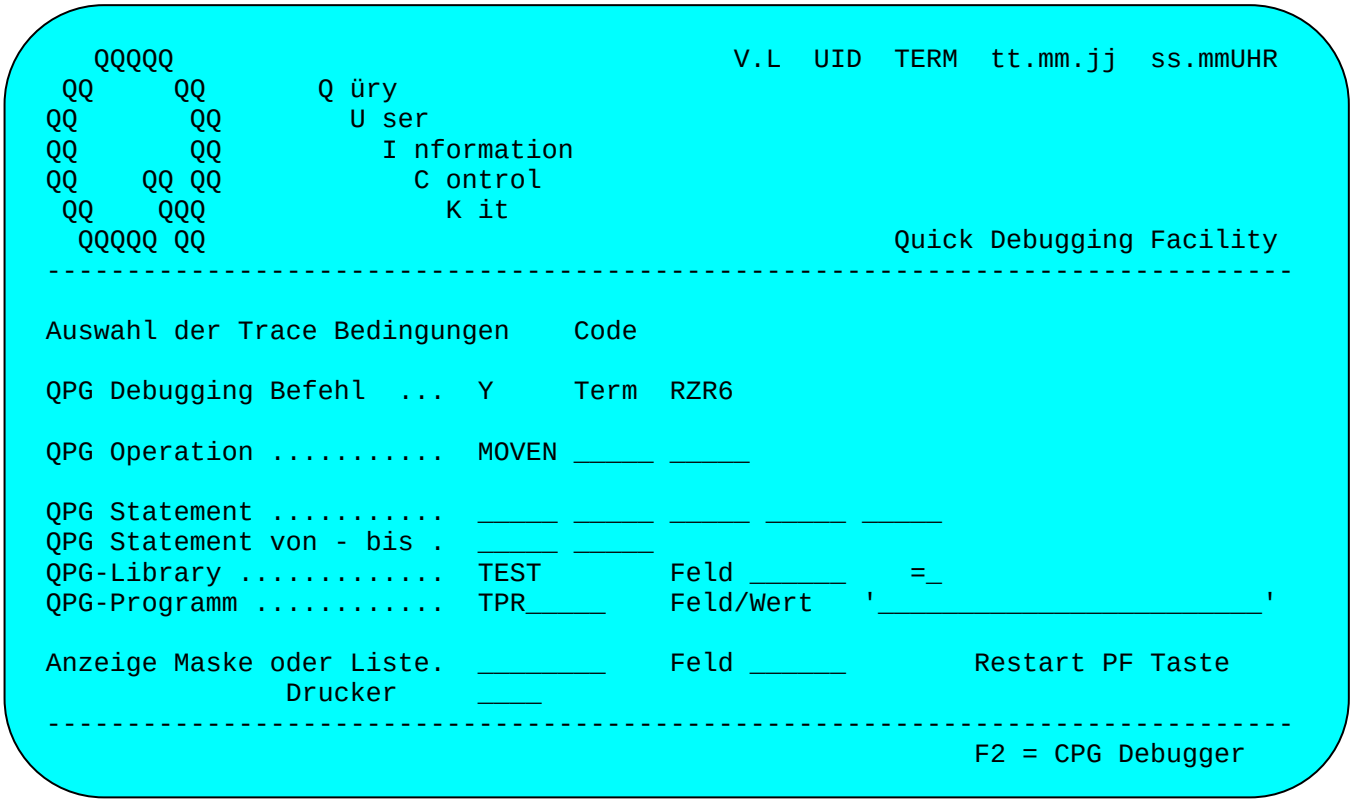
Für kompilierte oder bereits generierte QPG-Pgramme kann Debug auch mit der Transaktion QPG ein- und ausgeschaltet werden. Bei QPG sind als Befehle DEB ON (einschalten) oder DEB OFF (ausschalten) zusammen mit der Library und dem Programmnamen anzugeben, um den DEBUG zu aktivieren. Siehe auch Seite 5020.



Mit dem Befehl DEBug und der Option ON wird das Debug Facility auch dann wieder aktiviert, wenn es vorher mit PF3 ausgeschaltet wurde. Hiermit wird das DEBUG-Auswahlbild (erscheint, wenn keine Option angegeben wurde) übersprungen.

Es kann auch DEB mit einem 4-stelligen Code (bei Option) angegeben werden. Unter diesem Code bleiben dann die Trace-Bedingnugen, z.B. für Debug im Batch, gespeichert.

Wird als Befehl DEB ohne weitere Parameter oder mit einem CODE angegeben, dann erscheint das Bild zur Auswahl der Trace Conditions:



Die Parameter entsprechen dem Debug Facility QDF. Die Angaben Library und Programm erlauben jedoch zusätzlich noch die Auswahl, welche QPG-Programme getraced werden, wenn diese durch PROG Operationen miteinander verbunden sind.

Durch einen Eintrag bei Feld und Wert erreicht man, dass der DEBUG nur aktiviert wird, wenn das genannte Feld den angegebenen Wert hat. Solange das Feld einen anderen Wert hat, wird der DEBUG nicht aktiviert. Diese Funktion ist zur Zeit nur für alphanumerische Felder unterstützt.

Bei Term wird angegeben, an welchem Bildschirm DEBug aktiviert wird, Default ist der eigene Bildschirm.

Mit der Taste PF2 kann zusätzlich der CPG-Debugger QDF aufgerufen werden. Hiermit lässt sich der Test auf CPG-Programme und HL1-Module erweitern.

Ein Programm, das mit der Testhilfe abläuft, ermöglicht es, jedes Statement mit der interaktiven Testhilfe zu 'tracen'. Es erscheint folgendes Testbild:

```
QPGDPD      Online Debug Facility          V.L. UID  TERM  tt.mm.jj  11.53UHR
-------------------------------------------------------------------------------
Programm       :   TIRE       Library  :   PROG       Statement Nummer :
Transaction    :   QPG        Task-Nr. :   05087      Funktionstaste   :    DE
Modul          :   HPROG                              Restart PF-Key   :    PA
-------------------------------------------------------------------------------
Nach Operation :   START                               File Return Code :

Faktor1        :              ,        :
Faktor2        :              ,        :
                                       ....+....1....+....2....+....3..
Ergebnisfeld   :              ,        :
Service        :

Feld anzeigen  :   ______     , ___    :


                                       ....+....1....+....2....+....3..
-------------------------------------------------------------------------------
Enter: Weiter      PF4 : Trace Bedingungen PF8 : Feld anzeigen     Clear: Exit
PF1  : Hilfe       PF5 : Bildschirmdump    PF9 : Programm anzeigen
PF2  : User Bild   PF6 : Log Einzelsatz    PF10: Maske _________ anzg/dr: ____
PF3  : Ende Trace  PF7 : Log ohne Anzeige  PF11: Dump  ____      erstellen
```

Die Funktiostasten entsprechen (außer Taste PF9) dem Debug Facility (QDF).

Bei Maske oder Liste wird der Name einer Testmaske angegeben. Zusammen mit einem Druckernamen bei anzg/dr: wird jedoch der Name als der Name eines List-Dokumentes benutzt; hiermit wird ein Trace-Protokoll variabel gestaltet.

Sourcecode

Mit der Taste PF9 kann jetzt aus dem Testbild der Sourcecode des Programms angezeigt werden, falls dieser verfügbar ist. Bei der Anzeige wird das aktuelle Statement reversiv dargestellt. Mit Datenfreigabe oder PF8 wird vorwärts und mit PF7 rückwärts geblättert. Mit Enter wird das nächste Statement ausgeführt. In der Anzeige erscheint weiter der Sourcecode, wobei jetzt das nächste ausgeführte Statement markiert ist. F3 schaltet wieder zur normalen Debuganzeige zurück. Bei Start und Ende eines Programm erscheint ebenfalls die normale Debug-Anzeige.

Die Statement-Nummern werden ab der ersten Rechenbestimmung auf der rechten Seite angezeigt. Diese Statement-Nummern können dann z.B. in den Trace-Bedingungen ausgewählt werden. Stimmt das angezeigte Statement nicht mit der Operation im Trace-Bild überein, dann wurde der Sourcecode nach der letzten Programmumwandlung verändert und es wurde noch kein New-Copy ausgeführt. Dies sollte dann baldmöglichst nachgeholt werden, bzw. bei kompilierten QPG-Libraries sollte eine neue Umwandlung erfolgen, gegebenenfalls mit dem Parameter UPGRADE.

Wenn der Sourcecode nicht in der Datei QTFTXT gefunden wurde, dann sucht der Debugger ebenfalls in der Datei QTFARC.

Funktionen

Es können alle Felder mit Inhalt angezeigt werden. Feldgruppen werden mit allen Elementen angezeigt. Haben numerische Felder ungültige Inhalte, so erfolgt hierbei die Anzeige reversiv und in der definierten Fehlerfarbe. Hiermit lässt sich leicht prüfen, ob fehlerhafte Daten vorliegen, die bei Verarbeitung zu Abbrüchen führen können.

Bei den Stop-Conditions kann ein Feldinhalt auf >, <, =, >=, <= und >< abgefragt werden. Der Vergleich kann bei Alphafeldern mit einer maximal 24 Byte langen Konstanten erfolgen, die in Hochkommata eingeschlossen wird. Außerdem kann auch ein numerisches Feld abgefragt werden.

Wird bei Vergleich ein '*' angegeben (Scan), dann erfolgt ein Stop, wenn ein Alphafeld an einer beliebigen Stelle einen bestimmten Inhalt aufweist. Wird bei Vergleich '_*' angegeben (Not Scan) dann erfolgt der Stop, wenn das Alphafeld den gesuchten Inhalt nicht aufweist.

Bei den Stop-Conditions kann ein Feld mit einem anderen Feld verglichen werden. Ist das Stop-Condition-Feld oder das Vergleichsfeld nicht vorhanden, dann erfolgt kein Trace zu diesem Programm.

Wird bei Vergleich ein 'D' angegeben (Define), dann erfolgt ein Stop, wenn das Feld im Programm definiert ist.

Es kann ein Stop erfolgen, wenn ein Feld modifiziert wurde, dabei wird ein 'M' (Modify check) bei Vergleich angegeben. Bei Angabe eines Startwertes erfolgt der erste Stop, wenn das Feld hierzu einen unterschiedlichen Inhalt aufweist. Bei Alphafeldern werden für den Vergleich maximal 24 Stellen abgefragt.

Wird bei Vergleich ein 'R' angegeben (Reference), dann erfolgt ein Stop, wenn das Feld in einer Operation verwendet wird.

Wenn im Programm ein Indexfehler erkannt wird, so wird automatisch die Debug-Anzeige für das fehlerhafte Statement eingeblendet. Fehler lassen sich hierdurch wesentlich einfacher finden. Das gleiche gilt auch bei Division durch 0 und bei Divisionsüberlauf. Der falsche Index wird reversiv dargestellt (auch bei 0).

Bei Anzeige Einzelfeld (PF8) kann auch ein Feldgruppenelement mit Index angegeben werden. Außerdem wird die Feldanzeige automatisch aktualisiert.

Der Befehl DEBUG ON bleibt auch nach einer Operation PROGram wirksam.

Mit DEBUG wird das Ende eines Programms mit dem Hinweis ENDPR beim Operationscode angezeigt. ENDPR kann auch bei den Stop Conditions ausgewählt werden.

Bei der Auswahl der Trace-Conditions während der Ausführung kann auf ein anderes Programm umgeschaltet werden.

Felder und Konstanten werden bis zu einer maximalen Länge von 32 Stellen angezeigt.

Mit DEBUG kann auch die Operation MAP getestet werden. Wurde der Bildschirm durch DEBUG-Bilder verändert, so wird jetzt vor der MAP-Operation die ursprüngliche Bildschirmanzeige rekonstruiert, wobei jedoch die gewünschte Programmfunktionstaste erneut gedrückt werden muss.

Mit der Transaktion QPG und dem Befehl DEBug kann die Testhilfe auch an einem anderen Terminal ein- oder ausgeschaltet werden, indem die Terminal-Id entsprechend abgeändert wird. An dem anderen Terminal wird die Änderung wirksam, wenn dort entweder eine neue Task gestartet oder im Dialog eine MAP gelesen wurde. Die eigene Terminal-Id wird beim Befehl DEBug vorgegeben. Damit wird der Trace nur an dem betreffenden Terminal ein- oder ausgeschaltet. Wird die Terminal-Id gelöscht, dann wird der Trace für das betreffende Programm für alle Bildschirme ein- oder ausgeschaltet.

Bei dem Befehl DEBug können Programmname und Library sowohl im Startbild als auch im Eingabebild modifiziert werden. Enthält das Eingabebild andere Namen als das Startbild (wenn im Startbild die Namen nicht modifiziert wurden), dann erscheinen Library- und Programmname im Eingabebild (zur Kontrolle) reversiv.

Bei dem Befehl DEBug kann als Option außer ON und OFF ein CODE angegeben werden, um die Bedingungen unter diesem Code-Namen zu speichern und später einfach wieder abrufen zu können. Wird '?' als Code angegeben, dann wird ein Verzeichnis der gespeicherten Debug-Codes angezeigt. Wird als Debug-Befehl 'D' (delete) angegeben, dann werden die Debug-Conditions zu diesem Code gelöscht.

Die Debug-Funktion kann auch im Batch genutzt werden. Der Trace wird dabei als Protokoll im Logfile (TS-Queue QPGL, maximal 32000 Sätze) gespeichert. Das Protokoll wird erzeugt:

1. wenn im Programm OPTIONS DEBug angegeben wird.
2. bei der Operation DEBUG ON - (DEBUG OFF schaltet den Trace wieder ab).
3. mit dem Batch-Utility QPGUTIL und Angabe eines Trace-Code.

Der Logfile wird automatisch bei QPGUTIL gedruckt, oder - im HL1-Batch-Dump - falls ein Programmabbruch auftritt. In allen anderen Fällen kann der Log-File mit dem Serviceprogramm QPGLOG in der Library QPG gedruckt werden (z.B. als letzte Anweisung im Programmablauf).

Das Debug-Protokoll kann auch individuell mit Hilfe eines LIST-Dokuments gestaltet werden. Beim Erstellen des Protokolls wird bei MAP/LIST der Name des eigenen LIST-Dokumentes angegeben. Wird das Protokoll im CICS erstellt, dann ist dazu zusätzlich der Name eines Druckers erforderlich. Im Batch ist der Eintrag eines Druckers wahlweise. Der Drucker kann in den Trace-Bedingungen oder in der Debug-Anzeige eingegeben werden. Beim Drucken mit einem LIST-Dokument wird kein Logfile erstellt. Folgende LIST-Sections werden aufgerufen:

1. §SECTION OPCODE zum Drucken Opcode, Feldnamen, Statementnummer usw.
2. §SECTION FIELDS zum Drucken beliebiger Benutzerfelder.

Die Section OPCODE kann mit §INCLUDE QPGDL in die LISTE eingefügt werden.

Feldanzeige und Eingabe bei Debug

Im Debug-Bild kann mit PF8 oder PF10 auf die Full-Screen-Anzeige der Felder umgeschaltet werden, wenn kein Feld- bzw. Mapname angegeben wurde. Alle im Programm definierten Felder mit Inhalt werden in alphabetischer Reihenfolge angezeigt:

```
Debug Felder                                V.L  UID  T223  tt.mm.jj  14.18UHR
A24___ ___  _________________________
Feld   Wert                      Lng,D    Feld   Wert                      Lng,D
------ ----+----1----+----2---- -----    ------ ----+----1----+----2---- -----
A1                                 1      CPGVRL                        0    5,0
A10    XXXXXXXXXX                 10      C1                            1    1,0
A100   ************************ 100      C2                            2    2,0
A24                               24      C72                  12345,67-   7,2
B0          000                    8      FA            001               * 10
B159   0000000000?0?0?             15,9         2        002
CPGDAI                20000124    8,0          3        003
CPGDAT                24012000    8,0          4        004
CPGFRC                            2            5        005
CPGHPN TPR                        8            6        006
CPGHTM                           16            7        007
CPGMPF P3                         2            8        008
CPGMRC NI                         2            9        009
CPGPRC                            2           10        010
CPGPRL TEST                       4           11        011
CPGTCA                          256           12        012
CPGTID T223                       4           13        013
CPGTIM                  141821    6,0         14        014
------ ----+----1----+----2---- -----    ------ ----+----1----+----2---- -----
                           DE weiter    F3 return    F7 zurück
```

Bei Feldgruppen werden alle Elemente angezeigt. Mit Enter kann vorwärts geblättert werden und mit F7 wird wieder beim ersten Feld positioniert.

Positionieren

Im Kopf kann ein Feldname und bei Feldgruppen auch ein Index angegeben werden, ab dem die Anzeige positioniert wird. Es kann aber auch ein Feld mit dem Cursor ausgewählt, um den Feldinhalt zu ändern. Bei Enter wird der aktuelle Inhalt des Feldes im Kopf angezeigt.

Ändern von Feldinhalten

Nach Auswahl eines Feldes mit dem Cursor und Enter kann dessen Inhalt am Bildschirm verändert werden, sofern es kein geschütztes (internes) Feld ist. Im Kopf erscheint der Feldname (und -Index bei Feldgruppen) und das Feld kann verändert werden, ohne dass hierzu eine spezielle QSF-Maske erforderlich ist. Nach Enter wird die Anzeige ab dem ausgewählten Feld positioniert.

Kopieren von Feldinhalten

Beim Ändern von Feldinhalten kann der Feldname und der Index verändert werden. Handelt es sich beim alten und neuen Feld um Felder gleichen Typs (und bei numerischen Feldern mit gleicher Anzahl Dezimalstellen), dann wird der Feldinhalt auf das neue Feld kopiert. Dies gilt nicht bei geschützten (internen) Feldern. Beim Kopieren von Alphafeldern wird in der Länge des kürzeren Feldes kopiert.

Trace Conditions

Grundsätzlich erscheint das Testbild bei jedem Statement. Mit der PF4-Taste können die zu testenden Statements eingegrenzt werden. Es erscheint folgendes Bild:

```
  QQQQQ                                  V.L  UID  TERM  tt.mm.jj  ss.mmUHR
 QQ     QQ       Q üry
QQ       QQ        U ser
QQ       QQ          I nformation
QQ    QQ QQ            C ontrol
 QQ    QQQ               K it
   QQQQQ QQ                                          Quick Debugging Facility
--------------------------------------------------------------------------------

Auswahl der Trace Bedingungen

QPG Debugging Befehl  ...  Y     Term  R3R6

QPG Operation ...........  _____ _____ _____

QPG Statement ...........  _____ _____ _____ _____ _____
QPG Statement von - bis .  _____ _____
QPG-Library .............  ____        Feld  ______   =_
QPG-Programm ............  ________    Feld/Wert_____________________________

Anzeige Maske oder Liste.  ________    Feld  ______       Restart PF Taste __
               Drucker     ____
--------------------------------------------------------------------------------
```

Achtung: Bei einer neuen Transaction sind die Trace Conditions neu zu setzen.

Durch einen Eintrag bei Feld und Wert erreicht man, dass der DEBUG nur aktiviert wird, wenn das genannte Feld den angegebenen Wert hat. Solange das Feld einen anderen Wert hat, wird der DEBUG nicht aktiviert.

Diese Funktion ist z. Z. nur für alphanumerische Felder unterstützt.

Besondere Operationen

START wird bei Beginn des Programms aufgerufen. Hier können die übergebenen Feldinhalte überprüft und per QSF-Maske eingegeben werden.

ENDPR wird unmittelbar vor Rückkehr in das aufrufende Programm angezeigt.

Trace Log

Es kann auch mit PF6 ein Testprotokoll erstellt und mit PF7 auf die Testbilder verzichtet werden. Das Trace-Log liefert eine Zusammenfassung der protokollierten Statements. Es wird mit der Servicetransaktion QPG und der Auswahl LOG angezeigt und bei Bedarf gedruckt.

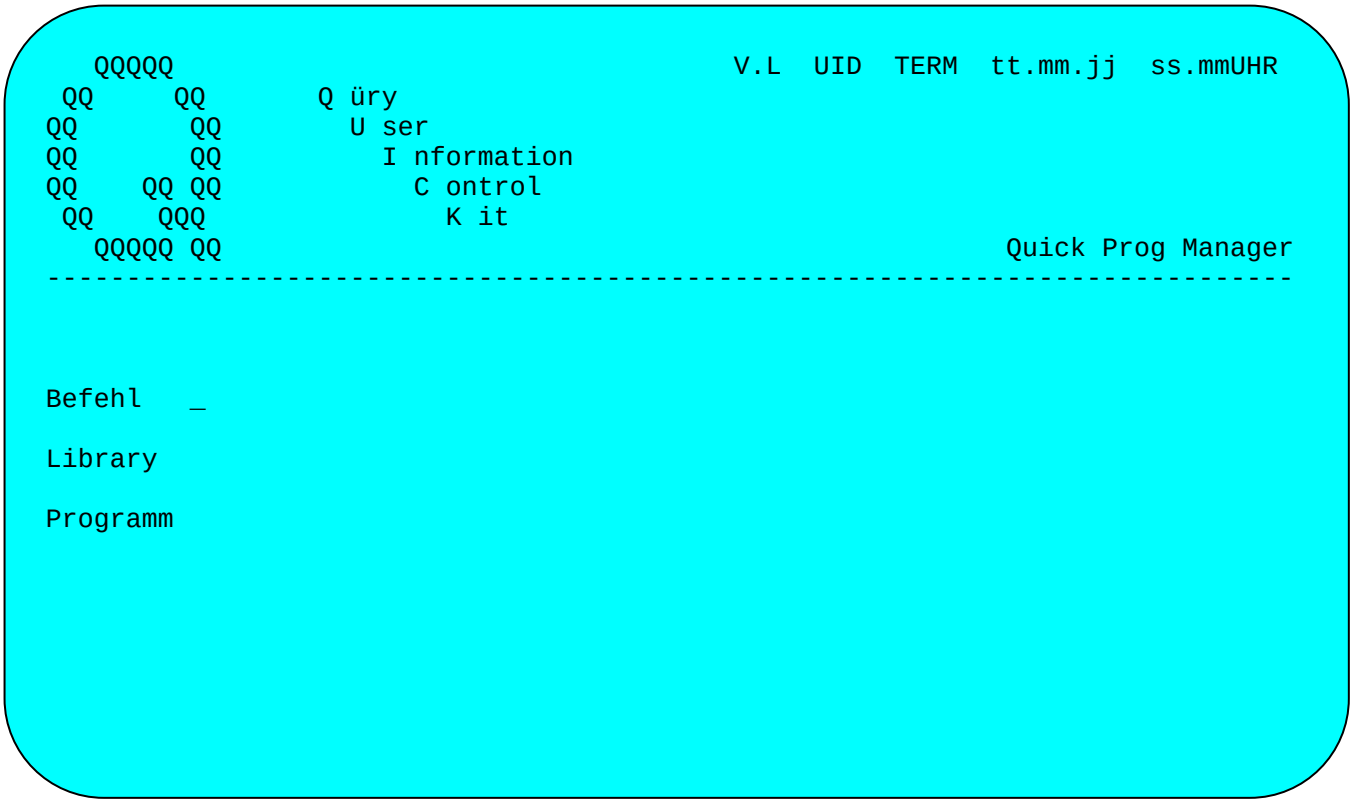
```
QPGDL Debug Logfile PROG TIRE              V.L  UID  TERM  tt.mm.jj  14.30UHR
                                                                      00020
opcod faktor1... .. faktor2... ergebnis.. ....+....1....+....2... pf Stmt lfnr
START                                                             DE       1
MOVEL                +          OP         +                     DE    1    2
MOVEL                F2         CPGMCU     F2                    DE    2    3
DO                                                                DE    3    4
MAPD                 PRTIRE                                       DE    4    5
IF    OP          EQ +                                           DE    5    6
+     ERG            F2         ERG                    10,00      DE    6    7
ELSE                                                              DE    7    8
END                                                               DE   24    9
=                         0  F2                        0,00      DE   25   10
END                                                               DE   26   11
DO                                                                DE    3   12
MAPD                 PRTIRE                                       DE    4   13
IF    OP          EQ +                                           DE    5   14
+     ERG            F2         ERG                    30,00      DE    6   15
ELSE                                                              DE    7   16
END                                                               DE   24   17
=                         0  F2                        0,00      DE   25   18
END                                                               DE   26   19
----- ------,--- -- ------,--- ------,--- ----+----1----+----2--- -- ---- ----
                                                        Drucken auf:
```

Beim Debuggen wird der Logbereich nach 10.000 Einträgen automatisch gerollt, so dass kein Überlauf eintritt und immer die letzten Einträge ausgewertet werden können.

Service-Transaktion QPG

Mit der Transaktion 'QPG' können Services für QPG-Programme aufgerufen werden, wie z.B. New-Copy für Programme oder Libraries.

Bei Aufruf von 'QPG' erscheint folgendes Bild:



Für das alternative Bildschirmformat steht die Service-Transaktion 'QPGA' zur Verfügung. Alle Befehle sind hier genau wie bei Transaktion 'QPG' unterstützt.

Mit der Taste PF1 werden die verfügbaren Befehle angezeigt, diese sind:

Befehl	Bedeutung
CLF	Clear Logfile
CLR	Löschen aller Pools
DEB	Debug Programm
DEL	Delete Programm
DIS	Display Programm
EDI	Edit Programm
GEN	Generiere Programm
NCO	Newcopy Programm/Library
NEXt	Aufruf nächstes Programm
LOG	Logfile anzeigen/drucken
PROG	Programm aufrufen
QTF	QTF-Aufruf
REF	Referen von Programmen
STA	Status Directory/Library
TASK	Programm aufrufen
TEST	Programm testen

Clear Logfile

CLF löscht ein mit DEBUG erstelltes Trace-Protokoll.

Löschen aller Pools

CLR erlaubt es, alle QPG-Libraries neu zu laden. Dieser Befehl sollte jedoch nur in Ausnahmefällen benutzt werden, da hierbei alle Benutzer beeinträchtigt werden können, die gerade QPG-Programme ausführen. Der Befehl NCO für einzelne Libraries ist nach Möglichkeit zu bevorzugen.

Diese Funktion ist nur für den System-Administrator bestimmt.

Debug-Programm

DEB aktiviert extern die Testhilfe mit DEBUG. Verschiedene Funktionen können hiermit aufgerufen werden (zusätzliche Parameter nach DEB):

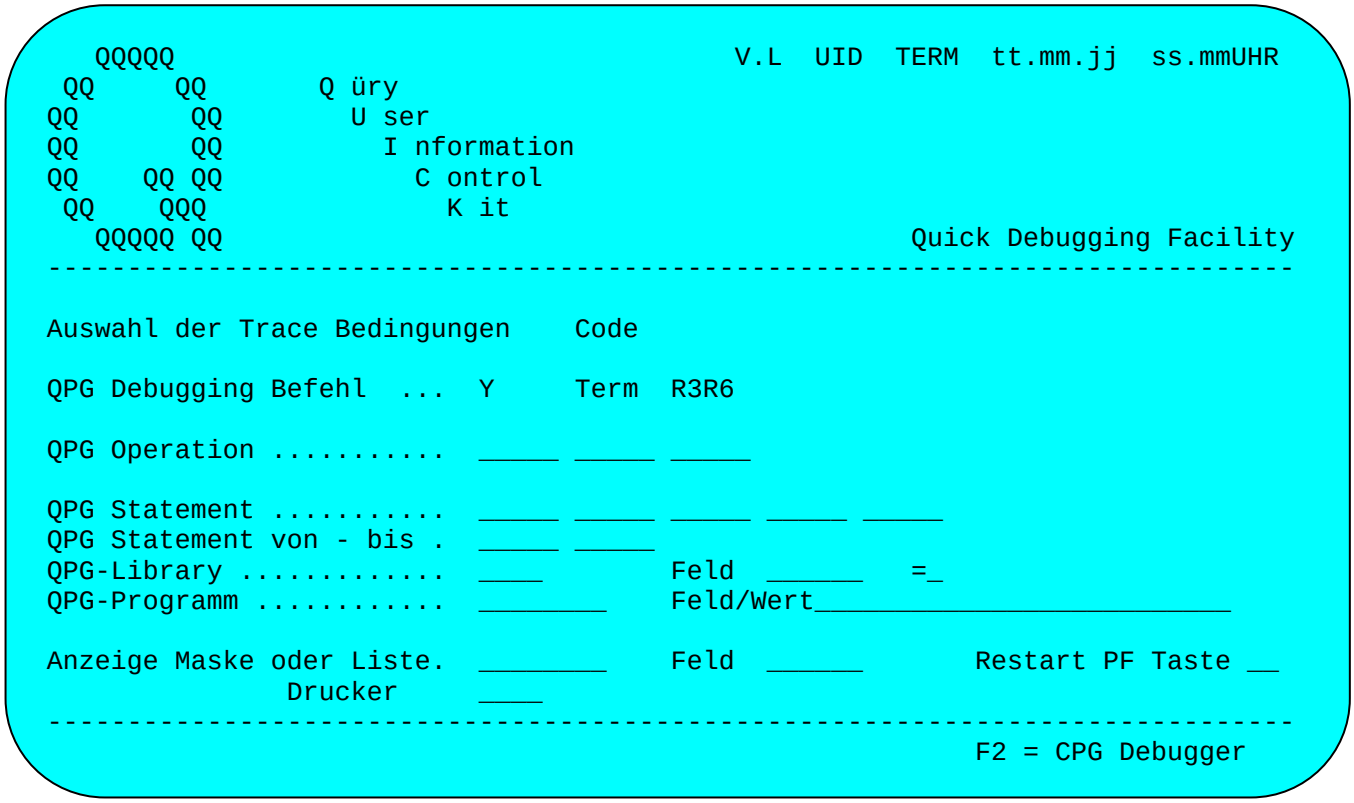
' ' Auswahl der Trace Conditions, siehe unten.

Diese Funktion kann auch erforderlich sein, wenn bereits ein DEBUG erfolgt ist und der Trace mit der Taste PF3 beendet wurde und keine Taste für den Restart des Trace verfügbar ist. Je nach Auswahl Y/N wird auch der Trace mit ON oder OFF ein- oder ausgeschaltet,wenn im Grundbild Library und Programm angegeben wurden.

'ON ' Einschalten des Trace-Facility.

'OFF' Ausschalten des Trace-Facility.

Wird als Befehl DEB ohne weitere Parameter angegeben werden, so erscheint die Auswahl der Trace Conditions:



Die Parameter entsprechen dem Debug Facility QDF. Die Angaben Library und Programm erlauben jedoch zusätzlich noch die Auswahl, welche QPG-Programme getraced werden, wenn diese durch PROG-Operationen miteinander verbunden sind.

Der Parameter Statement bis erfordert auch einen Eintrag bei Statement von.

Durch einen Eintrag bei Feld und Wert erreicht man, dass der DEBUG nur aktiviert wird, wenn das genannte Feld den angegebenen Wert hat. Solange das Feld einen anderen Wert hat, wird der DEBUG nicht aktiviert.

Bei Code wird angezeigt, ob und unter welchem Namen die Debug Bedingungen gespeichert werden.

Bei Feld kann ein Name angegeben werden, der als Stop-Condition benutzt wird. Hinter dem Feldnamen wird der Vergleich angegeben:

'= '	Abfrage auf gleich
'> '	Abfrage auf größer
'< '	Abfrage auf kleiner
'><'	Abfrage auf ungleich
'>='	Abfrage auf größer oder gleich
'<='	Abfrage auf kleiner oder gleich
'* '	Abfrage, ob Inhalt gefunden wurde (Scan)
'^*'	Abfrage, ob Inhalt nicht gefunden wurde (Not Scan)
'D '	Abfrage, ob Feld definiert ist
'M '	Abfrage, ob Feld modifiziert ist
'R '	Abfrage, ob Feld referenziert ist

Bei Feld/Wert wird das Vergleichsargument entweder als Feldname oder als Konstante (bei alpha in Hochkomma) angegeben. Bei Abfrage 'M' (modified) wird hier der Startwert, z.B. Blank vorgegeben. Bei jeder Feldänderung wird hier der Wert aktualisiert.

Bei Anzeige Maske oder Liste kann hier bereits der Name einer Testmaske oder eines List-Dokumentes sowie bei LIST ein Druckername vorgegeben werden.

Debug Codes

Wird der Befehl DEBug mit der Option '?' aufgerufen, dann werden alle gespeicherten Debug-Codes angezeigt:

```
QPGDVDC  QPG Verzeichnis Debug Codes        V.L  PR   RZR6  tt.mm.jj  17.17UHR
T8__
code y/n libr program  op1   op2   op3     st1   st2   st3   st4   st5   field
CODE  Y  TEST TPR      START       ENDPR
HUGO  Y  TEST TPR      =
TEST  Y  TEST
T1    Y  TEST TPR      START =     ENDPR                                   T1
T10   Y                -
T11   Y                /
T12   Y                MVR
T13   Y                SQRT
T14   Y                ADD
T15   Y                AFOOT
T17   Y                AAAAA BBBBB CCCCC
T18   Y
T2    Y                START =     ENDPR     2                             T1
T3    Y                START =     ENDPR     2     3                       T1
T4    Y                START =     ENDPR     2     3     4                 T1
T5    Y                START =     ENDPR     2     3     4     5           T1
T6    Y                START =     ENDPR     2     3     4     5     6     T1
T7    Y
T8    Y                *
```

In Zeile 2 kann ein Code angegeben werden, bei dem positioniert werden soll.

Delete Programm

DEL löscht ein bereits generiertes QPG-Programm aus der angegebenen Library. Anders als bei NCO erfolgt jedoch keine Prüfung, ob das Dokument in QTF noch gespeichert ist.

Newcopy Programm/Library

NCO löscht ein bereits generiertes QPG-Programm aus der angegebenen Library, damit beim nächsten Aufruf eine neue Kopie aus dem zugehörigen QTF-Programmdokument generiert werden kann. Existiert kein QTF-Programmdokument, so wird kein Newcopy ausgeführt.

Hinweis: Wird ein QPG-Programm aus dem QTF-Menü mit 'X' (eXecute) aufgerufen, so erfolgt durch Drücken der Taste PF2 automatisch ein Newcopy vor dem Programmaufruf. Ein Newcopy kann auch aus dem QTF-Menü mit 'N' aufgerufen werden, wenn der Benutzer in seinen Standardwerten erweiterte Funktionen QPG angegeben hat. Das gleiche gilt im QTF-Editor in Verbindung mit der Taste PF16.

Wird NCO angegeben ohne Programmname, dann wird ein New-Copy auf die komplette Library durchgeführt. Falls die Library neu kompiliert wurde und daher z.B. von CICS disabled wurde, so wird die Library durch NCO automatisch wieder verfügbar gemacht (enabled).

Programm aufrufen

NEXt ruft das nächste QPG-Programm in der angegebenen Library auf. Diese Funktion ist z. B. auch vorgesehen, um QPG-Programme über die Transaktion QPG taskorientiert aufzurufen.

Mit dem Befehl PROG kann ebenfalls in der Transaktion QPG ein Programm aufgerufen werden. PROG ist identisch mit dem Befehl NEXT, wobei jedoch zusätzlich bei taskorientierter Programmierung die Clear-Taste abgefragt werden kann.

Der Befehl TASK ist wie PROG als Erweiterung zu dem Befehle NEXT unterstützt. Bei TASK wird die nächste Transaktion mit der CLear-Taste beendet und außerdem die zuletzt mit TWA-SAVE erstelle TS-Queue gelöscht. Bei TASK werden in der Common Area CPGCOM die Stellen 21-24 für den TS-Namen benutzt.

Nach Ausführung des Befehls PROG, NEXT oder TASK wird wieder das QPG-Menü aufgerufen, wenn im QPG-Programm keine andere Steuerung erfolgt.

Aufruf per Programm mit Common Area

Alle Befehle können von 'außen' über die Common Area CPGCOM übergeben werden, wenn taskorientiert programmiert wird:

```
-C
        EDIT  CPGCOM
        EXITT 'QPG'
-O
        FIELD CPGCOM
                          4 'NEXT'
                  CPGPRL    8
                  CPGHPN   16
```

Die ersten Stellen der Common Area sind dabei für QPG reserviert:

```
CPGCOM   01 - 04   Befehl
         05 - 08   Library
         09 - 16   Programm
         17 - 20   Return Trans-Id
```

Die CPGPRL und CPGHPN sind interne Felder und werden von QPG mit dem Library- und dem Programmnamen gefüllt.

Aufruf per Programm mit Terminal Storage QPGA

Alle Befehle können auch in der TS-Queue QPGA übergeben werden, wenn QPG z.B. mit EXITI aufgerufen werden soll:

```
OPTIONS  DAT
         FILE QPGA
-D
         PRCMD               4.        * Command
         PROGN               8.        * PROGRAM NAME
         PRLIB               4.        * PROGRAM LIBRARY
         PTRID               4.        * Return Trans-id
-I
         FILE QPGA DD
-C
         READ QPGA
         PRCMD  = 'STA '
         PTRID  = 'MENU'
         WRITE QPGA
         EXITI 'QPG '
-O
         FILE QPGA DD
```

Die 32 Byte lange TS-Queue QPGA wird wie folgt im DD beschrieben:

QPGA	01 - 04	PRCMD	Befehl
	05 - 08	PRLIB	Library
	09 - 16	PROGN	Programm
	17 - 20	PTRID	Return Trans-Id
	21 - 32		reserviert

Logfile anzeigen/drucken

LOG ruft die Anzeige des Trace-Protokolls auf, sh. Kapitel 4000. Das Traceprotokoll erlaubt es, den Ablauf von Programmen aufzuzeichnen und diesen zu einem späteren Zeitpunkt mit LOG anzuzeigen oder zu drucken.

Status Directory/Library/Programm

STA zeigt eine QPG-Programmstatistik. Es gibt verschiedene Anzeigen, je nachdem, ob die Parameter Library und Programm angegeben werden:

DIRECTORY Status wird angezeigt, wenn nur STA ohne Library angegeben wird. Es werden standardmäßig nur die aktiven QPG-Libraries im CICS angezeigt. Wird die Option ALL verwendet, dann werden alle QPG-Libraries im CICS angezeigt. Mit dem Befehl STAT anstelle von STA wird immer zuerst die Statistik der Directory aufgerufen.

LIBRARY Status erfolgt bei Angabe der Library, wenn der Programmname fehlt. Es werden alle QPG-Programme in der Library angezeigt. In der Anzeige Status Library erfolgt jetzt ein Hinweis bei fehlerhaften Programmen. Ausserdem werden Programme, deren Source-Code nicht vorhanden ist, in einer anderen Farbe (türkis, normale Helligkeit) dargestellt.

PROGRAMM Status erfolgt bei Angabe von Library und Programmname. Es werden alle Felder des Programms angezeigt.

Es kann eine Option angegeben werden, die die Art der Anzeige steuert:

' '	siehe oben
'ALL'	Anzeige aller Libraries einschließlich der nicht aktiven
'DIR'	Anzeige der aktiven Libraries
'LIB'	Anzeige der Programme in der Library

Die folgenden Beispiele zeigen die verschiedenen Statistiken.

Status Directory

Wird der Command STA alleine angegeben, so wird eine Übersicht der von QPG benutzten Libraries angezeigt:

```
QPGDS    Quick Status Directory            V.L  UID  TERM  tt.mm.jj  16.34UHR

Lib  Kurzbeschreibung   Activ  erstellt / upgrade  Bytes belegt in %   Adresse
PROG Programme qpg        219  19.05.99 14:16 UPG 262320 205312   78  01DA4300
QPG  Programme qpg         48  14.12.99 15:36      93440  59904   64  01DE4400
TASK Transaktionen qpg      2                      65537    768    1  01E02B00
TEST Testprogramme qpg     10   8.04.98 14:34     131072   8192    6  01D79F80
QPGS Programme qpg          5  18.04.97 11:06      35152   2304    7  0080F008
PRG5 qpg lib für netpage   18   6.11.97  8:55      65537  21760   33  00774008
QXFS qxf services qpg     154  31.10.97 14:20 UPG 269728 222720   83  00730008
ODBC appc database access   0   2.07.99 13:11     131072      0    0  01CF7300
PRG1 Nicht gefunden         0   5.07.95 16:22      32768      0    0  01CEF280









Ende des Verzeichnisses.
```

Es werden die QPG-Libraries mit den im QTF benutzten Beschreibungen angezeigt, sowie Informationen zu den aktiven Libraries:

Lib	Name der Library und Kurzbescheibung, wenn verfügbar
Activ	Anzahl der Programme in der Library
erstellt	Datum und Uhrzeit des letzten COMPILE- oder CREATE-Laufs
upgrade	Hinweis, ob bei COMPILE der Parameter UPGRADE benutzt wurde
Bytes	Größe des Programm-Pools in der CICS PCT
belegt	hiervon benutzt in Bytes und in %
Adresse	Ladeadresse des Programm-Pools in der CICS PCT

Mit dem Cursor kann eine Library ausgewählt werden. Nach Enter wird dann die Statistik der Library angezeigt.

Die Tasten haben folgende Funktionen:

Enter	Vorwärts blättern, bei Ende zurück zum Startbild Verzweigen in Library Statistik mit Cursor Auswahl
F3	Zurück zum Startbild
F7	Rückwärts blättern

Status Library

Wird der Command STA zusammen mit einer Library angegeben, so wird eine Übersicht der von QPG in dieser Library benutzten Programme angezeigt:

```
QPGDSL   Quick Status Library PROG          V.L  UID  TERM  tt.mm.jj  16.34UHR

Programm Beschreibung                  Felder  PWA total Files  Bytes  Adresse
DSGEN    Dataset Generator                 63  348   356         1968 004E0018
LFILE    Logical files for dsgen           11  208   208          688 004E0818
UINFO    User Information                   8  216   216          236 004E0B18
HQTFA    QTF Aufruf                        13  276   276          332 004E0C18
HQTFK    Kopieren in QTF-Dokument           7  224   224          216 004E0E18
TIRE     Tischrechner                      23  196   200          632 004E0F18









Ende des Verzeichnisses.
```

Die Programme erscheinen in der Reihenfolge, in der sie in den Pool geladen wurden.

Es wird angezeigt, wie viele Felder im Programm definiert sind und wie viele Bytes diese in einer privaten Work Area (PWA) benötigen, sowie die Gesamtgröße der PWA einschließlich interner Felder. Es wird angezeigt, ob und wie viele Dateien ein Programm benutzt und auf welcher Ladeadresse es sich momentan im Pool befindet.

Mit dem Cursor kann ein Programm ausgewählt werden. Nach Enter wird dann die Statistik des Programms angezeigt.

Die Tasten haben folgende Funktionen:

Enter	Vorwärts blättern, bei Ende zurück zum Startbild Verzweigen in Programm Statistik mit Cursor Auswahl
F3	Zurück zum Startbild, bzw. zur Statistik der Directory
F7	Rückwärts blättern

Status Programm

Wird der Command STA zusammen mit einer Library und einem Programm angegeben, so wird die Übersicht der im QPG-Programm definierten Felder angezeigt:

```
qpgdsp   Quick Status   TEST TPR            V.L  UID  TERM  tt.mm.jj  15.55UHR
_______  _____
Feld   len,d  adr   Feld   len,d  adr   Feld   len,d  adr   Feld   len,d  adr
-----------------   -----------------   -----------------   -----------------
QOCK     1   726F   CPGHIC   4   625C   DDPCOD  10   62A5   CPGFRC   2   62F2
QFFTXT  14   7270   DDLIBR   4   6260   DDNAME   8   62AF   CPGHTM  16   62F4
QOCTXT  14   7270   DDFILE   8   6264   DDKART   1   62B7   CPGVRL   5,0 6304
AFAXNR  15   727E   DDTYPE   2   626C   DDKZ     1   62B8   SQCODE   5,0 6307
FAXNR   15   727E   DDIO     1   626E   DDIOMO   1   62B9   SQLCAF 120   630A
PS1      8,0 728D   DDFORM   1   626F   DDVAR    1   62BA   CPGHPN   8   6384
PS2      8,3 7292   DDBLEN   4,0 6270   DDSORT   1   62BB   UDATE    8   B820
DATCPG   6,0 7297   DDSLEN   4,0 6273   DDPROT   1   62BC   UTIME    8   B896
CPGJAH   2,0 729B   DDKLEN   2,0 6276   DDSYS    3   62BD   CPGTIM   6,0 B8B0
DATHL1   6,0 729D   DDART    1   6278   DDSTD    1   62C0   UDAY     2,0 B91F
HL1JAH   2,0 72A1   DDORG    1   6279   DDRDAT   8   62C1   UMONTH   2,0 B921
DATTOP   6,0 72A3   DDKPOS   4,0 627A   DDRSA    2   62C9   UYEAR    2,0 B923
TOPJAH   2,0 72A7   DDEINH   7   627D   DDFUDA   9,0 62CB   CPGDAT   8,0 B925
DATQTF   6,0 72A9   DDADD    1   6284   DDFTKZ   3   62D0   CPGPRC   2   B92A
BIGTWA*100   72AD   DDZYL    1   6285   DDDFCH   1   62D3   CPGMRC   2   B92E
A4       4   624D   DDAUSG   2,0 6286   DDPREF   2   62D4   CPGMPF   2   B930
A5       5   6251   DDPGR    3   6288   DDWSKZ   1   62D6   CPGPRL   4   B932
A6       6   6256   DDBESC  26   628B   RESFIL  27   62D7   UDATEC   2   B950
-----------------   -----------------   -----------------   -----------------
F2=Sort nach Name
```

Bei der Statistik eines Programms werden die Felder standardmäßig nach Adresse aufsteigend angezeigt. Mit der Taste PF2 kann die Sortierung geändert werden. Wird in der Startmaske die Option 'N' angegeben, so werden die Felder nach Name angezeigt.

Die Tasten haben folgende Funktionen:

Enter	Vorwärts blättern, bei Ende zurück zum Startbild
F2	Ändern der Sortierung
F3	Zurück zum Startbild, bzw. zur Statistik der Library
F7	Rückwärts blättern

Für die Anzeige können Startwerte vorgegeben werden. Dabei gilt bei:

Namen	Anzeige ab diesem Feld, sortiert nach Namen.
Adresse	Anzeige ab dieser Adresse, sortiert nach Adressen.

Beginnen mehrere aufeinander folgende Felder auf der gleichen Adresse, so wird hier die Anzeige besonders markiert (reversiv, türkis).

Die Adresse ist in der Form Bddd (B=Basisregister ddd=Displacement) angegeben. Dabei werden folgende Basisregister benutzt:

0	intern wie z.B. CPGCOM	
0	PWA	(Datenfelder des QPG-Programms, Bereich 28 bis 32 K-Bytes)
1	PWA	(Datenfelder des QPG-Programms, Bereich 24 bis 28 K-Bytes)
2	PWA	(Datenfelder des QPG-Programms, Bereich 20 bis 24 K-Bytes)
3	PWA	(Datenfelder des QPG-Programms, Bereich 16 bis 20 K-Bytes)
4	PWA	(Datenfelder des QPG-Programms, Bereich 12 bis 16 K-Bytes)
5	PWA	(Datenfelder des QPG-Programms, Bereich 8 bis 12 K-Bytes)
6	PWA	(Datenfelder des QPG-Programms, Bereich 4 bis 8 K-Bytes)
7	PWA	(Datenfelder des QPG-Programms, Bereich 0 bis 4 K-Bytes)
B	MBK	(Datenfelder in der Methodenbank)
C	TCA	(Datenfelder der Transaktion)

Programmtest

Der Befehl **TEST** erlaubt Progarmmtests an Masterterminals. Ist der Testschalter aktiviert, so wird an den definierten Masterterminals die jeweilige Programmlibrary durch eine Testlibrary ersetzt.

Wird der Befehl TEST ohne Aktion benutzt, so erscheint folgendes Bild zur Definition der Masterterminals:

```
QPGDT    Test Masterterminal ON            V.L  UID  R3R6  tt.mm.jj   8.46UHR

Library  Testlib  Terminal Programm Bis-Prog

PROG     TEST     RZR6     _________ _________

QXFS     QXFS     LW__     QXO______ QXP______

TASK     _____    R3R6     _________ _________

_____    _____    _____    _________ _________

_____    _____    _____    _________ _________

_____    _____    _____    _________ _________

_____    _____    _____    _________ _________

_____    _____    _____    _________ _________
```

Der Status ON oder OFF in Zeile 1 kann überschrieben werden, um die Testfunktion ein- oder auszuschalten.

Bei Library werden die QPG-Produktionslibraries angegeben, die an Masterterminals getestet werden sollen. Diese Libraries werden durch die unter Testlib angegebenen Testlibraries ersetzt. Der Test erfolgt an den unter Terminal angegebenen Masterterminals.

Soll nur ein bestimmter Bildschirm zum Testen benutzt werden, so muss dessen Terminal-Id vollständig angegeben werden. Wenn mit einer Gruppe von Terminals getestet wird, so wird nur der erste Teil (1-3 Stellen) der Terminal-Id für die jeweilige Terminalgruppe angegeben.

Es können mehrere Einträge für verschiedene Libraries oder Terminals angegeben werden.

Wird der Befehl TEST im QPG-Menü mit den Parametern ON oder OFF angegeben, so wird der Testschalter nur ein- oder ausgeschaltet, und die Anzeige der Masterterminals wird unterdrückt.

Die Testfunktion kann auf einzelne Programme (Eintrag bei Programm) und auf Gruppen von Programmen begrenzt werden (Einträge bei Programm und Bis-Prog).

Die Testfunktion kann auch benutzt werden, um eine Gruppe von Programmen an einem oder mehreren Bildschirmen mit Debug zu testen. Hierbei bleibt der Eintrag Testlib frei. Anschließend wird der Befehl DEBug aufgerufen, und bei der Auswahl der Trace-Bedingungen bleiben die Einträge Terminal-Id und QPG-Programm frei. Siehe Seite 5020.

QTF Aufrufe

Mit dem Befehl **DIS**play wird QTF aufgerufen und das QPG-Programm angezeigt.

Mit dem Befehl **EDI**t wird QTF aufgerufen, um das QPG-Programm zu verändern.

Mit dem Befehl **QTF** wird das Textsystem (Menü) aufgerufen.

Generieren Programm

Mit dem Befehl **GEN** wird ein Programmgenerator aufgerufen. Mit dem Generator können bestimmte Typen von Programmen generiert werden. Mit wenigen Eingaben lassen sich komplexe und funktionsfähige Programmobjekte erstellen, die dann bei Bedarf vom Programmierer verändert werden können. Folgende Prototypen sind zur Zeit unterstützt:

D	QPG-Datasets
H	HL1-Dataset für VSAM-Zugriffe
Q	Query-SQL-Dataset

Nach Eingabe des Befehls 'GEN' erscheint folgende Maske:

```
   QQQQQ                                  V.L  UID  TERM  tt.mm.jj  ss.mmUHR
 QQ     QQ       Q üry
QQ       QQ        U ser
QQ       QQ          I nformation
QQ    QQ QQ            C ontrol
 QQ    QQQ               K it
   QQQQQ QQ                                          Quick Programm Manager
--------------------------------------------------------------------------------

Generiere  _   Programm MUSTER__ Library TEST  Passwort          Ersetzen C


           D   QPG-Dataset

           H   HL1-Dataset

           O   Objekt Steuerprogramm

           Q   Query SQL Dataset



--------------------------------------------------------------------------------
```

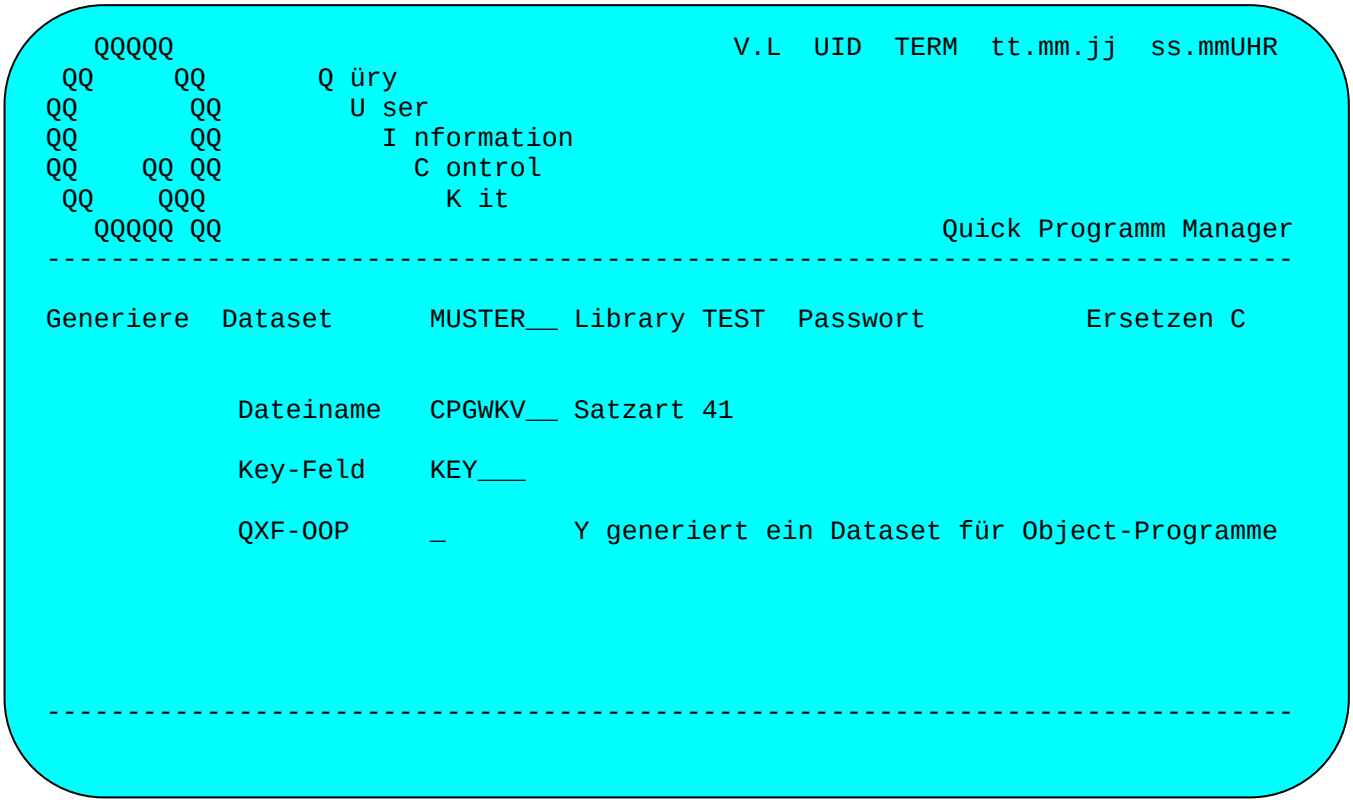
Hier wird bei Generiere der Typ des Programms ausgewählt, und es wird der Generator zum jeweiligen Programmtyp aufgerufen. Programm-Name, Library, Passwort und der Parameter Ersetzen können hier bereits vorgegeben werden. Für Ersetzen gilt:

'C'	Programm anlegen (Create). Fehlermeldung, wenn Programm existiert.
'X'	Wenn das Programm bereits existiert, wird es ersetzt.

Wird im QPG-Menü bei Befehl GEN bereits der Programmtyp als Option (D,H,O,Q) angegeben, so wird dieses Bild übersprungen, und es wird direkt der betreffende Generator aufgerufen.

Generieren QPG-Datasets

Mit 'D' werden QPG-Datasets generiert. Es erscheint folgendes Bild:



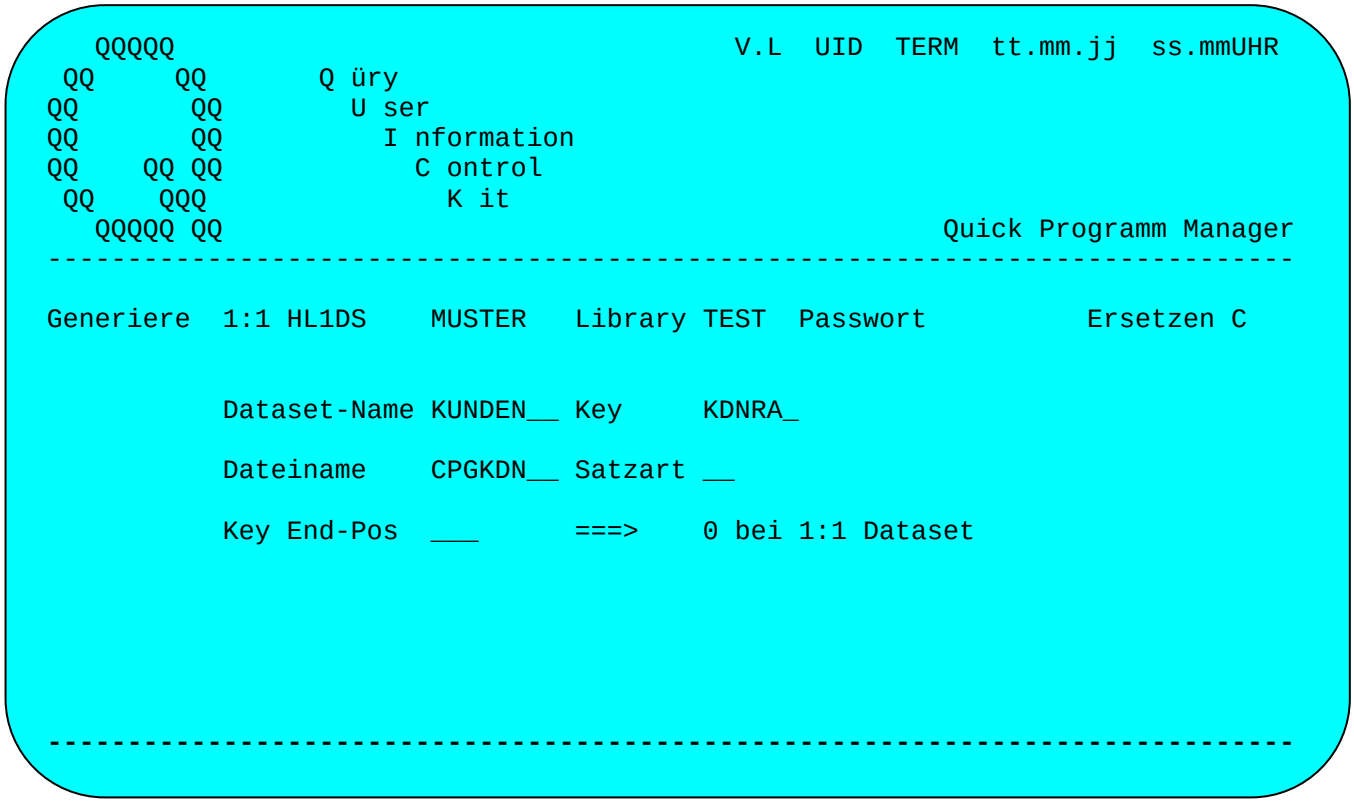
Aus den Eingaben wird folgendes Musterprogramm generiert:

```
*---------------------------------------------------------------------
*       QPG-Dataset                                          TEST.CPGWKV
*---------------------------------------------------------------------
options  define dataset
         file CPGWKV type 41.        * define file
-d.
         define CPGWKV type 41.      * define fields
              opcode         2.     * operation code

-i.
         file CPGWKV dd type 41.     * physical file
-c.
         opcode = cpgfrc.            * move operation code
         cpgfrc = '  '.              * init return code
         evaluate.
         when opcode = 'Z'.          * check
              check CPGWKV.
         when opcode = 'O'.          * open
              open CPGWKV.
         when opcode = 'G '.         * chain
              KEY chain CPGWKV.
         when opcode = 'GC'.         * chain - c
              KEY chain CPGWKV check.
         when opcode = 'GP'.         * chain - p
              KEY chain CPGWKV prot.
         when opcode = 'GU'.         * chain - u
              KEY chain CPGWKV upd.
         when opcode = 'S'.          * setll
              KEY setll CPGWKV.
         when opcode = 'R'.          * read
              read CPGWKV.
         when opcode = 'B'.          * readb
              readb CPGWKV.
         when opcode = 'D'.          * rndom
              rndom CPGWKV.
         when opcode = 'U'.          * updat
              updat CPGWKV.
         when opcode = 'N'.          * write
              write CPGWKV.
         when opcode = 'L'.          * delet
              KEY delet CPGWKV.
         when opcode = 'C'.          * close
              close CPGWKV.
         end-evaluate
```

Generieren HL1-Datasets

Mit 'H' werden HL1-Datasets generiert. Es erscheint folgendes Bild:



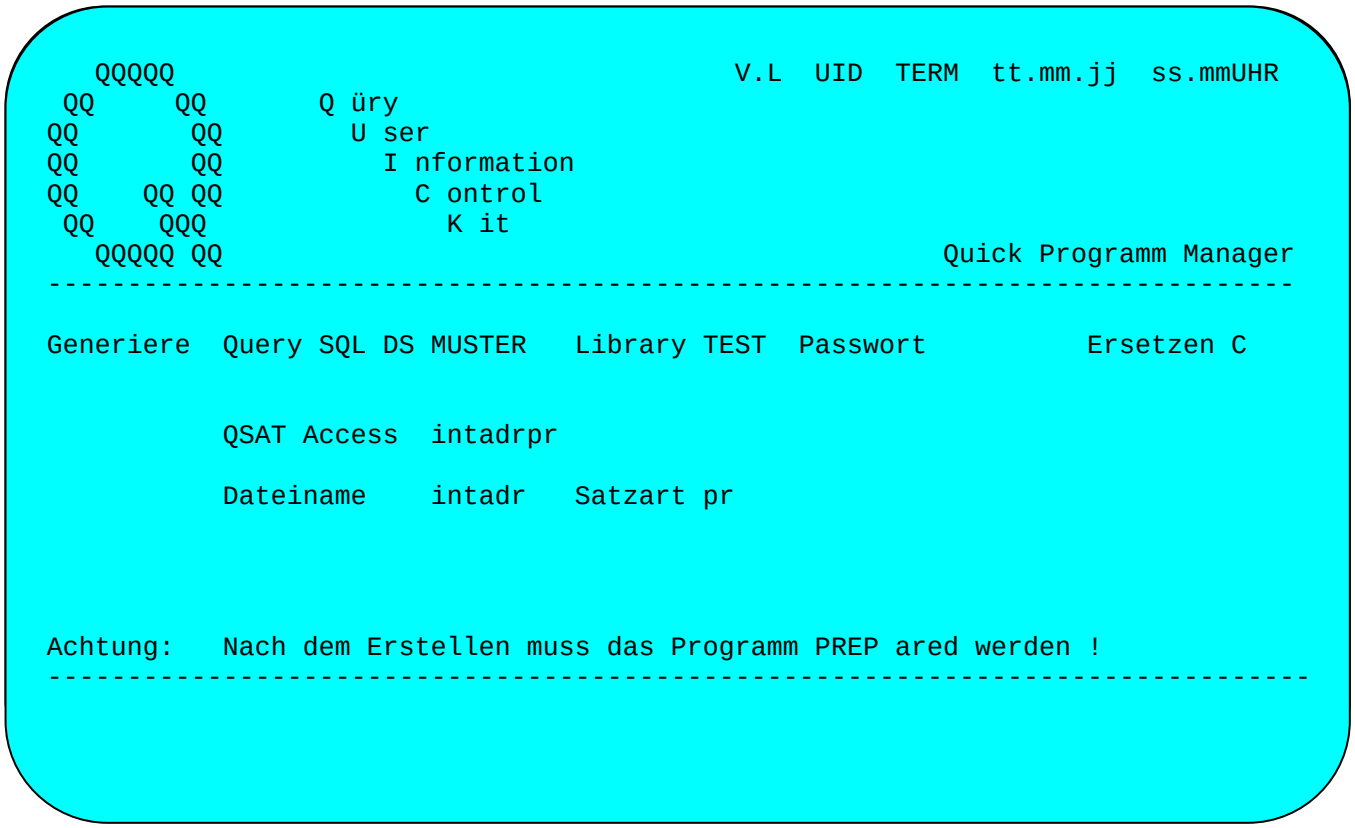
Mit diesen Eingaben wird folgendes HL1-Dataset generiert:

```
*-------------------------------------------------------------
* HL1-Dataset                                      TEST.MUSTER
*-------------------------------------------------------------
OPTIONS DIC PWA TIT DATASET PHA KUNDEN.
   FILE CPGKDN
-D.
   DEFINE KUNDEN
   OC                2.        * OPERATION CODE
   RC                1.        * RETURN CODE
-I.
   FILE CPGKDN DD.                          * PHYSICAL FILE
-C.
   OC = CPGHIC.                             * SAVE OPERATION CODE
   RC = ' '.                                * INIT RETURN CODE
   IF OC = 'Z'.                             * CHECK
      CHECK CPGKDN
      IF CPGFRC = 'N'.
         RC = 'G'                           * NOT OPEN
      END.
   END
   IF OC = 'O'.                             * OPEN
      OPEN CPGKDN
      IF CPGFRC = 'N'.
         RC = 'F'                           * NOT OPEN
      END.
   END
   IF OC = 'G '.                            * CHAIN
      KDNRA CHAIN CPGKDN
   END
   IF OC = 'GC'.                            *  CHAIN - C
      KDNRA CHAIN CPGKDN CHECK
   END
   IF OC = 'GP'.                            *  CHAIN - P
      KDNRA CHAIN CPGKDN PROT
   END
   IF OC = 'GU'.                            *  CHAIN - U
      KDNRA CHAIN CPGKDN UPD
   END
   IF OC = 'G' AND.
   IF CPGFRC = 'NF'.
      RC = 'G'.                             * NOT FOUND
   END
   IF OC = 'R'.                             * READ
      KDNRA READ CPGKDN
      IF CPGFRC = 'EF'.
         GETHS
      END
   END
   IF OC = 'B'.                             * READB
      KDNRA READB CPGKDN
      IF CPGFRC = 'EF'.
         GETHS
         RC = 'E'
      END
   END
```

```
      IF OC = 'D'.                          * RNDOM
         RNDOM CPGKDN
      END
      IF OC = 'U'.                          * UPDAT
         EXCPT UPDAT
      END
      IF OC = 'N'.                          * WRITE
         EXCPT WRITE
         IF CPGFRC = 'DR'.
            RC = 'D'                        * DUPLICATE RECORD
         END.
      END
      IF OC = 'L'.                          * DELET
         KDNRA DELET CPGKDN
      END
      IF OC = 'C'.                          * CLOSE
         CLOSE CPGKDN
         IF CPGFRC = 'N'.
            RC = 'F'                        * NOT CLOSED
         END.
      END
      MOVE RC TO CPGHIC.                    * MOVE RETURN CODE
   -O.
      FILE CPGKDN DD ALG UPDAT.             * UPDATE
      FILE CPGKDN DD ADD WRITE.             * WRITE
```

Generieren Query-SQL-Dataset

Mit 'Q' werden Query-SQL-Datasets generiert. Es erscheint folgendes Bild:



Mit den Eingaben wird folgendes Programm generiert:

```
*-------------------------------------------------------------------
*       qpg Query sql dataset                              TEST.INTADR
*-------------------------------------------------------------------
options  dataset.
data division.
  define INTADR type PR
         opcode              1.        * operation code
         cpgcur              4.        * cursor
         wherev             64.        * where variable
procedure division.
         opcode = cpgfrc.
         evaluate.
         when opcode = 'O' or.          * batch open
         when opcode = 'S'.             * online setll
              prog sqlbeg prog.
§qsat         access INTADRPR
              if cpgtid > ' '.          * only in cics
                 mapo qpg$psw.
                 cpgcur = '2423'.
                 prog cpgcur qpg.       * set cursor
                 mapi qpg$psw low.      * get where conditions
              end.
              sql wherevar :wherev
         when opcode = 'R'.             * read batch and cics
§qsat         fetch INTADRPR
         when opcode = 'D' or.          * online rndom
         when opcode = 'C'.             * batch close
              sql close cursor.
         end-evaluate.
```

Das Dataset enthält Anweisungen für den QPG-Preprocessor. Diese (Macros) werden aufgelöst, indem das Programm mit QPG PREP (sh. Preprocessor Seite 1900) preprocessed wird. Nach dem Preprocess sieht das Programm wie folgt aus:

```
*-------------------------------------------------------------------
*       qpg Query sql dataset                              PROG.INTADR
*-------------------------------------------------------------------
options  dataset.
data division.
  define INTADR type PR
         opcode              1.        * operation code
         cpgcur              4.        * cursor
         wherev             64.        * where variable
procedure division.
         opcode = cpgfrc.
         evaluate.
         when opcode = 'O' or.         * batch open
         when opcode = 'S'.            * online setll
              prog sqlbeg prog.
*qsat         access INTADRPR
              SQL ACCESS                                              +
                  A.ASS,                                              +
                  A.CICS,                                             +
                  A.COBOL,                                            +
                  A.CPG,                                              +
                  A.CSP,                                              +
                  A.DB2,                                              +
                  A.DL1,                                              +
                  A.FAX,                                              +
                  A.FIRMA,                                            +
                  A.FIRMA2,                                           +
                  A.HODERF,                                           +
                  A.KDNRA,                                            +
                  A.KURZN,                                            +
                  A.MASCH1,                                           +
                  A.MVS,                                              +
                  A.NAME,                                             +
                  A.ORT,                                              +
                  A.PLZ,                                              +
                  A.POSTF,                                            +
                  A.RPG,                                              +
                  A.SAP,                                              +
                  A.SQLDS,                                            +
                  A.STR,                                              +
                  A.TEL,                                              +
                  A.VM,                                               +
                  A.VSEESA,                                           +
                  A.VSESP                                             +
                  FROM                                                +
                     SQLDBA.INTADR A
              if cpgtid > ' '.         * only in cics
                 mapo qpg$psw.
                 cpgcur = '2423'.
                 prog cpgcur qpg.      * set cursor
                 mapi qpg$psw low.     * get where conditions
              end.
              sql wherevar :wherev
```

```
         when opcode = 'R'.            * read batch and cics
*qsat         fetch INTADRPR
              SQL FETCH CURSOR INTO                                    +
                  :ASS,                                                +
                  :CICS,                                               +
                  :COBOL,                                              +
                  :CPG,                                                +
                  :CSP,                                                +
                  :DB2,                                                +
                  :DL1,                                                +
                  :FAX,                                                +
                  :FIRMA,                                              +
                  :FIRMA2,                                             +
                  :HODERF,                                             +
                  :KDNRA,                                              +
                  :KURZN,                                              +
                  :MASCH1,                                             +
                  :MVS,                                                +
                  :NAME,                                               +
                  :ORT,                                                +
                  :PLZ,                                                +
                  :POSTF,                                              +
                  :RPG,                                                +
                  :SAP,                                                +
                  :SQLDS,                                              +
                  :STR,                                                +
                  :TEL,                                                +
                  :VM,                                                 +
                  :VSEESA,                                             +
                  :VSESP
         when opcode = 'D' or.         * online rndom
         when opcode = 'C'.            * batch close
              sql close cursor.
         end-evaluate.
```

Wird dieses Dataset als FILE im Data Dictionary angelegt (Org='P' und Library 'TEST'), dann kann es mit Query (sh. CPG3 Serviceprogramme) bearbeitet werden.

Referenz von Programmen

Mit dem Befehl **REF** kann im QPG online eine Statistik pro Library erstellt werden. Diese Statistik liefert die Referenzen der aktiven Programme nach folgenden Kriterien:

C Operationen
D Data Dictionary
F Dateien
H Module
K Konstanten
L List Dokumente
M Masken
N Feldnamen
P Programme
T Trans-id's

Anhand eines eingegebenen Suchargumentes wird hierzu der Verwendungsnachweis erstellt. Beim Suchen nach Dateinamen werden z.B. alle (aktiven) Programme der Library angezeigt, in denen die angegebene Datei definiert ist.

Bei der Reference kann auch generisch gesucht werden, um z.B. ein Feld, eine Datei oder eine Konstante, die mit einem bestimmten Namen oder Zeichenfolge beginnt, aufzufinden. Dabei wird das Suchargument mit '*' beendet.

Außerdem liefert die Statistik auch eine Anzeige aller Objekte eines bestimmten Typs, wenn kein Suchargument eingegeben wurde.

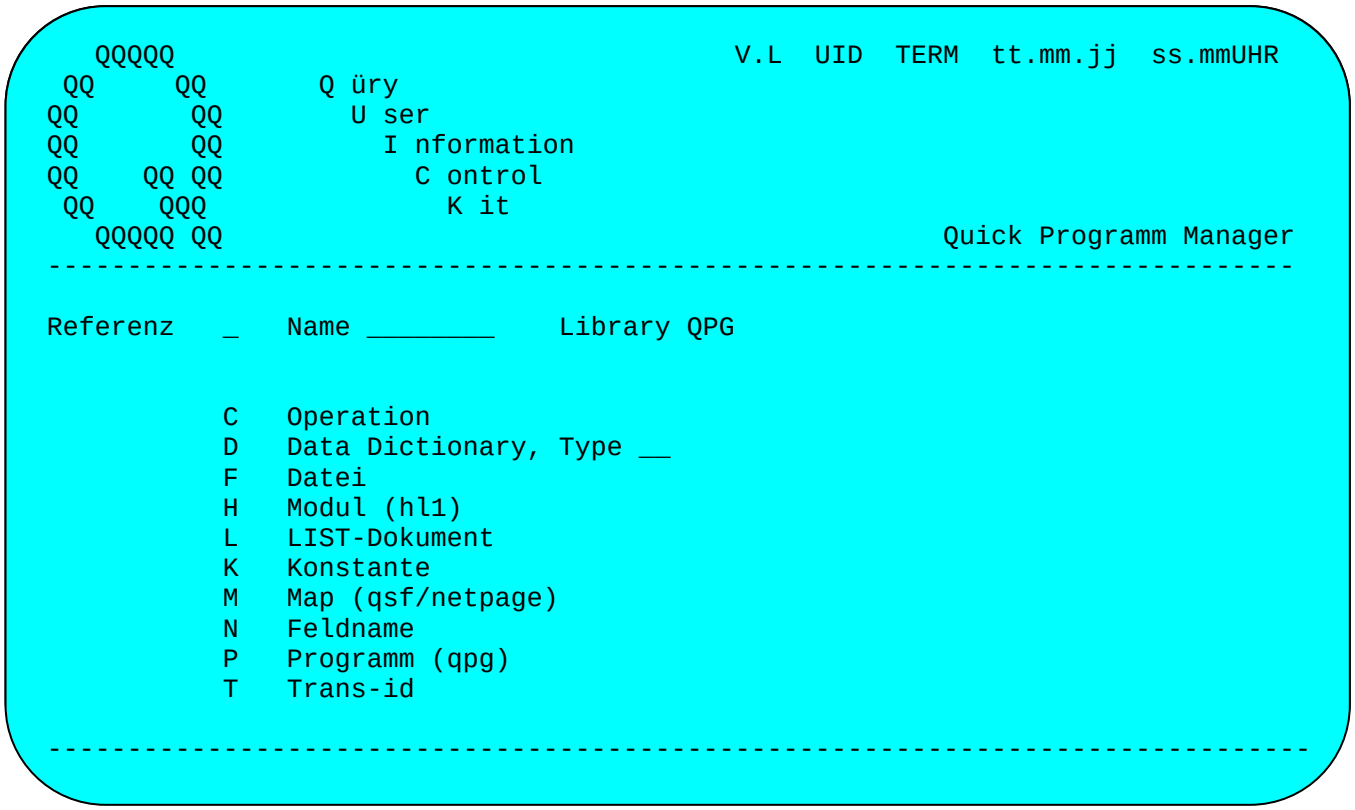
Es werden nur die Programme in der Library durchsucht, deren Object-Code verfügbar ist (aktive Programme). Bei kompilierten Libraries sind in der Regel alle Programme aktiv, wogegen bei nicht kompilierten Libraries (z.B. in der Testumgebung) nur die benutzten Programme aktiv sind. Programme, die nach einem Newcopy nicht wieder aufgerufen wurden, sind von der Statistik ausgegeschlossen.

Sortierung

In der Statistik sind die Programme nach Ihrer Reihenfolge beim Laden in die Library geordnet. Bei kompilierten Libraries ist daher die Reihenfolge i.d.R. alphabetisch, wogegen bei Testlibraries die Reihenfolge der Aufrufe entscheidend ist. Um eine Reference von allen Programmen zu erhalten, ist auch in der Entwicklungsumgebung ein Kompilieren der Libraries (mit Option UPGRADE) zu empfehlen.

Aufruf der Referenz von Programmen

Wird als Befehl REF angegeben, um eine bestimmte Library zu durchsuchen, dann wird das Auswahlbild für die Programmreference aufgerufen:



In diesem Menü kann jetzt ausgewählt werden, nach welchen Suchargument die Reference erstellt wird. Durch Auswahl (z.B. mit 'C' für Operationen) wird die Suche nach Datenfreigabe gestartet. Mit Taste PF3 erfolgt der Rücksprung in das (Grund)-Menü.

Wird ein Suchargument angegeben, dann werden nur die Programme angezeigt, in denen das betreffende Objekt vorkommt, z.B. werden beim Suchen nach Dateinamen werden alle (aktiven) Programme der Library angezeigt, in denen die angegebene Datei definiert ist.

Wird kein Suchargument angegeben, dann werden alle Objekte des ausgewählten Typs angezeigt, die in den aktiven Programmen vorkommen. Die Reihenfolge der Objekte in den Programmen entspricht ihrer Reihenfolge im generierten Objektcode.

Die Suche erfolgt nur, wenn die Library geladen ist. Falls die Library noch nicht benutzt wurde, kann sie in QPG mit dem Befehl NCO (Newcopy) aktiviert werden.

Referenz bei Operationen

Bei Auswahl 'C' erfolgt eine Anzeige nach Operationen. Wird z.B. in der Library QPG nach der Operation SQL gesucht, dann erscheint folgende Anzeige:

```
Reference C  SQL          in Lib QPG       V.L  PR   TR93  tt.mm.jj  10.50UHR

Programm Beschreibung                      Objekte
PREPQSAT Prepare qsat access
PREPSQLD Prepare QPG-Program
SQLHELPA Anzeige sql help text









Ende des Verzeichnisses.
```

Hinweis: Die Beschreibung wird nur angezeigt, wenn außer dem Objektcode auch der Sourcecode für die Library im QTF verfügbar ist.

Wird kein Suchargument angegeben, dann werden alle Operationen in ihrer Reihenfolge angezeigt.

Referenz bei Data Dictionary

Bei Auswahl 'D' erfolgt eine Anzeige nach Data Dictionary.

Wird z.B. in der Library QXFS nach der DD Struktur QXIA, Satzart (Type) PR gesucht, dann erscheint folgende Anzeige:

```
Reference D  QXIA     PR  in Lib QXFS       V.L  PR   TR93  tt.mm.jj  11.09UHR

Programm Beschreibung                       Objekte
OCPRDREF Objektprogramm Dic Ref
OCPRKLBS Objektprogramm KLBS
OCPRNOFL Objektprogramm no file
OCPRXDIC Objektprogramm Dic
QXCDIC   Default Modell Object Control
QXCKLB   Default Modell Object Control
QXCOPL   Object Control
QXCREF   Object Control
QXOOHF   Object Control
TESTPRO  Default Modell Object Control









Ende des Verzeichnisses.
```

Es werden alle aktiven Programme angezeigt, in denen die gesuchte DD-Struktur verwendet wurde.

Wird bei Auswahl 'D' kein Suchargument angegeben, so werden alle verwendeten Data Dictionaries in den Programmen angezeigt:

```
Reference D                in Lib QXFS       V.L  PR   TR93  tt.mm.jj  11.14UHR

Programm Beschreibung                        Objekte
EXIFQTFI Exit QTF Index                      QXSA        QXSA
EXIFQTFL Exit QTF Index                      QXSA        QXSA
EXIFQTFW Exit QTF Index                      QXSA        QXSA
FPPSDDST Funktion Input Page DD Str          H2DD1    DS DP
IODSDREF QPG-Dataset - qxf oop               QXFDRF
IODSKLBS DatasetKomunikation QXFOLB          QXFKLB
IODSXDIC DatasetKomunikation QXFOLB          QXFDIC
IOPGQ2DD q2 Data Dictionary Seite            HDDFIL      HDDFLL      TITLE
                                             HEADER      DP
OCPRDREF Objektprogramm Dic Ref              QXFDRF      QXOP        QXFS
                                             QXSA        QXIA     PR QXFD
                                             QXFC        QXTX        QXDG
OCPRKLBS Objektprogramm KLBS                 QXFKLB      QXOP        QXFS
                                             QXSA        QXIA     PR QXFD
                                             QXFC        QXTX        QXDG
OCPRNOFL Objektprogramm no file              QXOP        QXFS        QXSA
                                             QXIA     PR QXFD        QXFC
                                             QXTX        QXDG
OCPRXDIC Objektprogramm Dic                  QXFDIC      QXOP        QXFS
                                             QXSA        QXIA     PR QXFD
```

Mit Datenfreigabe kann bis zum Ende geblättert werden. Mit F7 kann die Suche ab dem ersten Programm neu gestartet werden.

Wird ein generisches Suchargument angegeben, dann werden alle Programme angezeigt, die Data Dictionaries enthalten, die mit dem Suchargument beginnen.

```
Reference D  HQTF*        in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.14UHR

Programm Beschreibung                       Objekte
PREPO    Prepare QPG-Program only macro     HQTFC
QPGDC    QTF-Dokumentenverzeichnis          HQTFF
QPGGEN   Generate Modell                    HQTFC
QTFA     QTF Aufruf                         HQTFA    PR
QTFADMI  define qtf administrators          HQTFA
QTFAV    QTF Anzeige Verzeichnis            HQTFA
QTFAVA   QTF Anzeige Vergleich Archiv       HQTFC
QTFC     QTF dataset                        HQTFC
QTFE     QTF extended editor commands       HQTFA
QTFK     Kopieren in QTF-Dokument           HQTFK
QTFLC    QTF Library Check                  HQTFF
QTFLIST  QTF-Dokument auflisten             HQTFC
QTFLV    QTF Library Verfalldatum           HQTFF
QTFSORTS QTF-Dokument sortieren             HQTFC
REF      global program reference           HQTFA
TXTA     QTF Aufruf - mit TX Namen          HQTFA    TX
TXTC     QTF Dataset - mit TX Namen         HQTFC    TX
TXTDOC   QTF Text Dataset                   HQTFC    TX


Ende des Verzeichnisses.
```

Mit Datenfreigabe kann bis zum Ende geblättert werden. Mit F7 kann die Suche ab dem ersten Programm neu gestartet werden.

Referenz bei Dateien

Bei Auswahl 'F' erfolgt eine Anzeige nach Dateien. Wird z.B. in der Library QPG nach der Datei HQTFC gesucht, dann erscheint folgende Anzeige:

```
Reference F  HQTFC        in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.17UHR

Programm Beschreibung                       Objekte
PREP     Prepare QPG-Program
QPGGEN   Generate Modell
QTFAVA   QTF Anzeige Vergleich Archiv
QTFC     QTF dataset
QTFLIST  QTF-Dokument auflisten
QTFSORTS QTF-Dokument sortieren
TXTC     QTF Dataset - mit TX Namen









Ende des Verzeichnisses.
```

Wird kein Suchargument angegeben, dann werden alle Dateien angezeigt:

```
Reference F                in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.21UHR

Programm Beschreibung                        Objekte
PREP     Prepare QPG-Program                 HQTFC    QTFS
QPGA     QPG Aufruf                          QPGA
QPGC     QPG Call                            QPGA     QPGC
QPGDC    QTF-Dokumentenverzeichnis           HQTFF    QPGV
QPGGEN   Generate Modell                     HQTFC    QTFS     QPGA
QPGLOG   QPG print debug logfile             QPGL
QTFA     QTF Aufruf                          HQTFA
QTFAV    QTF Anzeige Verzeichnis             QPGA     QPGW     HQTFA
QTFAVA   QTF Anzeige Vergleich Archiv        HQTFC    QTFS
QTFAVB   QTF Anzeige Vgl Archiv Blätter      QTFS
QTFC     QTF dataset                         HQTFC
QTFDV    QTF Drucken Verzeichnis             QPGV
QTFE     QTF extended editor commands        QSFZ     TPTC     HQTFA
QTFLC    QTF Library Check                   HQTFF
QTFLIST  QTF-Dokument auflisten              HQTFC
QTFLV    QTF Library Verfalldatum            HQTFF    QPGV
QTFSORTS QTF-Dokument sortieren              HQTFC    QTFS     HSORT
QUERY    Query Aufruf                        TPQA
QXFA     QXF Aufruf                          QXSA
SQLHELP  display sql help text               QPGA
```

Wird ein generisches Suchargument angegeben, dann werden alle Programme angezeigt, die Dateien enthalten, die mit dem Suchargument beginnen.

```
Reference F  MAYDA*       in Lib PROG       V.L  PR   TR93  tt.mm.jj  11.14UHR

Programm Beschreibung                       Objekte
MAYAR000 Matchcode-Auswahl Artikel          MAYDAI
MAYAR001 Matchcode Artikelstamm Name        MAYDAIA
MAYAR002 Matchcode Artikelstamm Kuenstl     MAYDAIB
MAYAR003 Matchcode Artikelstamm Pseudon     MAYDAIC
MAYAR004 Matchcode Artikelstamm Bildart     MAYDAID
MAYAR005 Matchcode Artikelstamm Preisgr     MAYDAIE
MAYAR006 Matchcode Artikelstamm Blattfo     MAYDAIF
MAYAR007 Matchcode Artikelstamm Bild-Fo     MAYDAIG
MAYAR008 Matchcode Artikelstamm Technik     MAYDAIH
MAYAR020 Artikelstammanzeige über Start     MAYDAI
MAYAR021 Artikelstammanzeige über Hyper     MAYDAI
MAYAR025 Kundenstamm - prüfen von - bis     MAYDAI
MAYKD004 Kunden-Artikel anzeigen            MAYDAI
MAYTEST  Artikelstammanzeige über Hyper     MAYDAI
MAYTEST2 Kundenstamm - prüfen von - bis     MAYDAI
MAYUM001 Umsatz-Anzeige Kunde des Artik     MAYDAI
MAYUM002 Umsatz-Anzeige über Kundeninfo     MAYDAI
MAYUM010 Kunden-Artikelumsätze berechne     MAYDAI
MAYUM020 UMSATZ-BERECHNUNG FüR ARTIKELS     MAYDAI

Ende des Verzeichnisses.
```

Referenz bei Modulen

Bei Auswahl 'H' erfolgt eine Anzeige nach HL1-Modulen. Wird z.B. in der Library QPG nach dem Modul HQTFA gesucht, dann erscheint folgende Anzeige:

```
Reference H  HQTFA        in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.29UHR

Programm Beschreibung                      Objekte
QTFA     QTF Aufruf
QTFAV    QTF Anzeige Verzeichnis
QTFE     QTF extended editor commands
TXTA     QTF Aufruf - mit TX Namen









Ende des Verzeichnisses.
```

Hinweis: HL1-Module werden auch dann angezeigt, wenn das Modul als HL1-Dataset in der File Section definiert wurde. (Datasets werden nicht mit EXHM, sondern z.B. mit READ, CHAIN, UPDATE u.s.w. aufgerufen).

Wird kein Suchargument angegeben, dann werden alle HL1-Module angezeigt:

```
Reference H                in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.40UHR

Programm Beschreibung                        Objekte
CICSLNK  exec cics link - expr with no       HPRCP
CICSNCO  Newcopy of resident cics progr      HPRCP    HPRCP    HPRCP    HPRCP
CPGCUR   Cursor setzen                       HCURS
HELP     Help Aufruf                         HHELP
PREP     Prepare QPG-Program                 HQTFC
QPGCHECK check qpg object code               HPRNC    HPREP
QPGDC    QTF-Dokumentenverzeichnis           HQTFF
QPGGEN   Generate Modell                     HQTFC
QPGNCOPY QPG Newcopy                         HQLFA    HPRNC
QPGPREP  prepare new qpg object code         HPRNC    HPREP
QTFA     QTF Aufruf                          HQTFA
QTFAV    QTF Anzeige Verzeichnis             HQTFA    HQLFA
QTFAVA   QTF Anzeige Vergleich Archiv        HQTFC
QTFC     QTF dataset                         HQTFC    HQTFC
QTFE     QTF extended editor commands        HQTFA    HQTFA
QTFK     Kopieren in QTF-Dokument            HQTFK
QTFLC    QTF Library Check                   HQTFF
QTFLIST  QTF-Dokument auflisten              HQTFC
QTFLV    QTF Library Verfalldatum            HQTFF
QTFSORTS QTF-Dokument sortieren              HQTFC    HSORT
```

Referenz bei Konstanten

Bei Auswahl 'K' erfolgt eine Anzeige nach Konstanten. Wird z.B. nach der Konstanten 'NF' in der Library QPG gesucht, dann erscheint folgendes Bild:

```
Referenz  K  NF          in Lib QPG       V.L  PR   T222  tt.mm.jj  12.25UHR

Programm Beschreibung                     Objekte
CHKAPPC  check appc connection
CPGSTA   CPG/RPG Statistik
QSFLOAD  load QSF-Maps
QTFLC    QTF Library Check









Ende des Verzeichnisses.
```

Wird kein Suchargument angegeben, dann werden alle Konstanten angezeigt:

```
Reference K                in Lib QPG        V.L  PR   T222  tt.mm.jj  12.14UHR

Programm Beschreibung                       Objekte
CHKAPPC  check appc connection              0                NOT
                                            INS              ACQ
                                            NF               connection not fo
                                            NI               connection not in
                                            NA               connection not ac
                                            CEMT INQ CONN(    )
CHKAPPCA check appc connection abend         P1               P2
                                            QTF              P3
                                            P4               ON
                                            CPGAPPC ABEND of
CICSLNK  exec cics link - expr with no      EXPR
CICSNCO  Newcopy of resident cics progr     ENA              EXPR
                                            DFHEMTP          PE
                                            NEW              EXPR
                                            DFHEMTP          PC
                                            CEMT SET PROG(    )
CPGMCU   set map cursor                     FIND
CPGMFN   get long fieldname                 SCAN
CPGTIPX  TCP-IP Explorer Interface          R                D
                                            N                $TCP
```

Wird ein generisches Suchargument angegeben, dann werden alle Programme angezeigt, die Konstanten enthalten, die mit dem Suchargument beginnen.

```
Reference K  PASSW*       in Lib PROG       V.L  PR   TR93  tt.mm.jj  11.14UHR

Programm Beschreibung                       Objekte
ANMELDU  ANGELEGT VOM PC                    Passwort ist fals Passwort fehlt









Ende des Verzeichnisses.
```

Referenz bei LIST-Dokumenten

Bei Auswahl 'L' erfolgt eine Anzeige nach QTF-LIST-Dokumenten. Wird z.B. in der Library QPG nach QTF$V gesucht ($=Sprachencode), dann erscheint folgende Anzeige:

```
Reference L  QTF$V        in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.47UHR

Programm Beschreibung                      Objekte
QTFDV    QTF Drucken Verzeichnis




Ende des Verzeichnisses.
```

Wird kein Suchargument angegeben, dann werden alle LIST-Dokumente angezeigt:

```
Reference L                in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.48UHR

Programm Beschreibung                        Objekte
QPGLOG   QPG print debug logfile             QPGDL    QPGDL    QPGDL    QPGDL
QTFDV    QTF Drucken Verzeichnis             QTF$V    QTF$V    QTF$V
QTFLIST  QTF-Dokument auflisten              QPGDL    QPGDL    QPGDL




Ende des Verzeichnisses.
```

Referenz bei Masken

Bei Auswahl 'M' erfolgt eine Anzeige nach QSF-Masken. Wird z.B. in der Library QXFS nach QXDDPAGE gesucht, dann erscheint folgende Anzeige:

```
Reference M  QXDDPAGE     in Lib QXFS       V.L  PR   TR93  tt.mm.jj  11.53UHR

Programm Beschreibung                       Objekte
FPPSDDST Funktion Input Page DD Str
IOPGQ2DD q2 Data Dictionary Seite
QXFIPG   Funktion Input Page QXDPAG
QXMDPG   Page of File DIC
QXMKPG   Page of File KLB
QXMNPG   Page of File OLB
QXMSPG   Page of File OSL









Ende des Verzeichnisses.
```

Wird kein Suchargument angegeben, dann werden alle QSF-Masken angezeigt:

```
Reference M                in Lib QXFS       V.L  PR   TR93  tt.mm.jj  11.55UHR

Programm Beschreibung                        Objekte
FPPSDDST Funktion Input Page DD Str          QXDDPAGE
IOPGQ2DD q2 Data Dictionary Seite            QXDDPAGE
OTWBOBJD QXF OOP Werkbank                    QXMDOLBR QXFERR   QXFERR   QXMDWBCH
OTXPEXEC OSP: Checkpoint abfragen.           QXFERR
QTFDOK   Seitenanzeige QTF                   QXDQTF   QXDQTF
QTFDOK2  Seitenanzeige QTF                   QXDQTF   QXDQTF
QTFTXT2  Seitenanzeige QTF mit Dokument      QXDQTF   QXDQTF
QTMDIR   Seitenanzeige QTF                   QXDQTF
QTMTXT   Seitenanzeige QTF                   QXDQTF   QXDQTF
QXFGRO   Page of File OLB                    QXDDSTAT
QXFIPG   Funktion Input Page QXDPAG          QXDDPAGE
QXIASH   Archiv set history                  QXMDOLBS
QXIASL   Archiv set lock                     QXMDOLBS
QXIASR   Archiv set release                  QXMDOLBS
QXIASV   Archiv set version                  QXMDOLBS
QXITGG   Input Page Transfer Group Get       QXDDGROP
QXITGR   Funktion Input Page QXDPAG          QXDDGROP
QXITGS   Funktion Input Page QXDPAG          QXDDGROP
QXITOS   Trasfer ObjProg to Save Lib         QXMDOLBS
QXITSO   Trasfer Save to ObjProg Lib         QXMDOLBS
```

Referenz bei Feldnamen

Bei Auswahl 'N' erfolgt eine Anzeige nach QPG Feldnamen. Wird z.B. in der Library QPG nach dem Feld DOKUM gesucht, dann erscheint folgende Anzeige:

```
Reference N  DOKUM        in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.57UHR

Programm Beschreibung                       Objekte
PREP     Prepare QPG-Program
QPGDC    QTF-Dokumentenverzeichnis
QPGGEN   Generate Modell
QTFA     QTF Aufruf
QTFAV    QTF Anzeige Verzeichnis
QTFAVA   QTF Anzeige Vergleich Archiv
QTFAVB   QTF Anzeige Vgl Archiv Blätter
QTFC     QTF dataset
QTFDV    QTF Drucken Verzeichnis
QTFE     QTF extended editor commands
QTFK     Kopieren in QTF-Dokument
QTFLIST  QTF-Dokument auflisten
QTFSORT  QTF-Dokument sortieren
QTFSORTS QTF-Dokument sortieren




Ende des Verzeichnisses.
```

Es kann auch nach dem im Data Dictionary gespeicherten Langnamen gesucht werden.

Wird ein generisches Suchargument angegeben, dann werden alle Programme angezeigt, die Felder enthalten, die mit dem Suchargument beginnen.

```
Reference N  SQ*          in Lib QPG        V.L  PR   TR93  tt.mm.jj  11.14UHR

Programm Beschreibung                       Objekte
CHKAPPCA check appc connection abend        SQCODE SQLCAF SQLISL
CPGSTA   CPG/RPG Statistik                  SQCODE SQLCAF SQLISL
CPGSTAR  CPG/RPG Statistik Reference        SQCODE SQLCAF SQLISL
DDREF    data dictionary reference          SQCODE SQLCAF SQLISL
PREP     Prepare QPG-Program                SQCODE SQLCAF SQLISL
PREPO    Prepare QPG-Program only macro     SQCODE SQLCAF SQLISL
PREPQTF  Prepare text statements            SQCODE SQLCAF SQLISL
PREPSQL  Prepare qsat/SQL-Statements        SQCODE SQLCAF SQLISL
PREPSQLT Prepare qsat/sql stmts menue       SQCODE SQLCAF SQLISL
PREPTEXT Prepare text statements            SQCODE SQLCAF SQLISL
QPGC     QPG Call                           SQCODE SQLCAF SQLISL
QPGDC    QTF-Dokumentenverzeichnis          SQCODE SQLCAF SQLISL
QPGGEN   Generate Modell                    SQCODE SQLCAF SQLISL
QPGLOG   QPG print debug logfile            SQCODE SQLCAF SQLISL
QTFADMI  define qtf administrators          SQCODE SQLCAF SQLISL
QTFAV    QTF Anzeige Verzeichnis            SQCODE SQLCAF SQLISL
QTFAVA   QTF Anzeige Vergleich Archiv       SQCODE SQLCAF SQLISL
QTFAVB   QTF Anzeige Vgl Archiv Blätter     SQCODE SQLCAF SQLISL
QTFDV    QTF Drucken Verzeichnis            SQCODE SQLCAF SQLISL
QTFE     QTF extended editor commands       SQCODE SQLCAF SQLISL
```

Referenz bei Programmen

Bei Auswahl 'P' erfolgt eine Anzeige nach QPG-Programmen. Wird z.B. in der Library QPG nach dem Programm QTFA gesucht, dann erscheint folgende Anzeige:

```
Reference P  QTFA         in Lib QPG        V.L  PR   TR93  tt.mm.jj  12.01UHR

Programm Beschreibung                      Objekte
QPGGEN   Generate Modell
QTFAV    QTF Anzeige Verzeichnis
QTFE     QTF extended editor commands









Ende des Verzeichnisses.
```

Hinweis: QPG-Programme werden auch dann angezeigt, wenn das Programm als QPG-Dataset in der File Section definiert wurde. (QPG-Datasets werden nicht mit PROG oder TASK, sondern z.B. mit READ, CHAIN, UPDATE u.s.w. aufgerufen).

Wird kein Suchargument angegeben, dann werden alle QPG-Programme angezeigt:

```
Reference P                in Lib QPG        V.L  PR   TR93  tt.mm.jj  12.09UHR

Programm Beschreibung                       Objekte
HELPA1   Help Aufruf mit PA1                HELP
HELPF1   Help Aufruf mit PF1                HELP
PREP     Prepare QPG-Program                TXTA     TXTA     PREPS
PREPQTF  Prepare text statements            PREPTEXT
PREPSQL  Prepare qsat/SQL-Statements        PREPSQLT PREPSP   SQLBEG
QPGGEN   Generate Modell                    QPGA     QTFK     QTFA     QTFA
                                            QPGGEN
QTFAV    QTF Anzeige Verzeichnis            QPGC     QTFA     QTFAVA   QPGNCOPY
                                            PREP     QTFA     QTFA     QTFA
QTFAVA   QTF Anzeige Vergleich Archiv       QTFAV    QTFAV    QTFAVB
QTFAVB   QTF Anzeige Vgl Archiv Blätter     QTFAV    QTFAVB
QTFDV    QTF Drucken Verzeichnis            QPGDC
QTFE     QTF extended editor commands       QTFA     QPGC     PREPS    QTFA
                                            QTFA     QTFA     QTFA     QPGC
                                            QTFA     QTFA     QTFA
QTFLV    QTF Library Verfalldatum           QTFLC
QTFSORT  QTF-Dokument sortieren             QTFSORTS
SQLHELP  display sql help text              SQLHELPA
SQLHELPA Anzeige sql help text              SQLHELP  SQLHELP  SQLBEG   SQLHELP

Ende des Verzeichnisses.
```

Referenz bei Trans-Ids

Bei Auswahl 'T' erfolgt eine Anzeige nach CICS-Trans-Ids. Wird z.B. in der Library QPG nach der Trans-Id QTF gesucht, dann erscheint folgende Anzeige:

```
Reference T  QTF          in Lib QPG       V.L  PR   TR93  tt.mm.jj  12.12UHR

Programm Beschreibung                      Objekte
QTFA     QTF Aufruf
SQLHELP  display sql help text
TXTA     QTF Aufruf - mit TX Namen









Ende des Verzeichnisses.
```

Wird kein Suchargument angegeben, dann werden alle QPG-Programme angezeigt, in denen die Operationen EXITI und EXITT verwendet wurden:

```
Reference T              in Lib QPG        V.L  PR   TR93  tt.mm.jj  12.13UHR

Programm Beschreibung                      Objekte
QPGA     QPG Aufruf                        QPG
QPGGEN   Generate Modell
QTFA     QTF Aufruf                        QTFF     QTFE     QTF
QUERY    Query Aufruf                      TPQO
QXFA     QXF Aufruf                        QXF
SQLHELP  display sql help text             QTF
TASK     task control
TXTA     QTF Aufruf - mit TX Namen         QTFF     QTFE     QTF









Ende des Verzeichnisses.
```

Hinweis: Es werden hier auch die Programme angezeigt, in denen z.B. EXITI mit einer variablen Trans-Id oder EXITT ' ' (Return nach CICS) codiert wurde. In diesen Fällen wird kein Objektname angezeigt.

Service-Library QPG

Mit QPG wird die interne Programmlibrary QPG für Servicefunktionen zur Verfügung gestellt. Diese Library enthält genormte QPG-Programme, die in bestehende QPG-Anwendungen integriert werden können.

Folgende Programme sind verfügbar:

CICSLNK	Exec CICS Link - EXPR ohne TWA
CICSNCO	Newcopy residenter CICS-Programme
CPGCUR	Setzen des Cursors auf den Bildschirm
HELPA1	Aufruf der Masken bezogenen Hilfe mit PA1
HELPF1	Aufruf der Masken bezogenen Hilfe mit PF1
PREP	Preprocessor für QPG-Sourcecode
PREPO	Preprocessor für QPG-Sourcecode
QPCF	Dataset für komprimierte Ausgabe von QPCF-Daten
QPCFXL	wie QPCF, aber mit 999 Bytes Satzlänge.
QPGA	Aufruf der QPG-Services.
QPGCHECK	Check QPG Object Code
QPGDC	Dokumentenverzeichnis
QPGLOG	Druckt das Debug-Protokoll aus.
QPGNCOPY	QPG und QLF Newcopy
QPGPREP	Compile QPG-Programm.
QPGSTRT	Start weiteres QPG-Programm
QTFA	Aufruf des Textsystems.
QTFC	Textdataset
QTFDV	Drucken Verzeichnis
QTFK	Kopieren in QTF-Dokument
QTFLIST	QTF-Dokument auflisten
QTFLOAD	QTF-Dokument laden
QTFLV	Library Verfalldatum
QTFSCAN	Durchsuchen einer Textlibrary nach einem Stichwort
QTFSCANS	Durchsuchen einer Textlibrary nach einem Stichwort (Subroutine)
QTFSORT	QTF-Dokument sortieren
QTFSORTS	QTF-Dokument sortieren (Subroutine)
QUERY	Aufruf des Report-Generators
QXFA	Aufruf des Expertensystems
TASK	Taskorientierter Programmstart
TXTA	Aufruf des Textsystems. Wie QTFA, jedoch mit TX-Feldnamen.
TXTC	Textdataset. Wie QTFC, jedoch mit TX-Feldnamen.
TXTH	Aufruf Help Facility mit TX-Feldnamen.

Beispiel

Mit dem Befehl:

PROGRAM QTFSORT QPG

wird das Programm QTFSORT in der Servicelibrary QPG aufgerufen, mit dem z.B. ein bestehendes QTF-Dokument sortiert werden kann (sh. Seite 5540). Wichtig für den erfolgreichen Aufruf eines QPG-Serviceprogramms ist die korrekte Übergabe der benutzten Daten. Hierzu müssen die Feldnamen so angegeben werden, wie diese vom QPG-Serviceprogramm vorgeschrieben werden. Muss-Felder sind immer anzugeben und zu füllen, Kann-Felder sind nur bei Bedarf anzugeben. Nach Rückkehr sind gegebenenfalls Returncodes abzufragen.

CICSLNK Exec CICS Link - EXPR ohne TWA

Hierzu wird im Feld PROGN das betreffende Programm angegeben.

```
DATA DIVISION

PROGN               8.      * PROGRAM NAME

PROCEDURE DIVISION

EDIT CPGCOM.                * COMMUNICATION AREA
PROGN = 'DFHEMTP '.         * CICS COMMAND PROCESSOR
PROG CICSLNK QPG.           * EXPR WITH NO TWA

OUTPUT DIVISION.

FIELD CPGCOM.
                    'CEMT SET TER(T223) IN SERVICE '
```

CICSNCO Newcopy residenter CICS-Programme

Hierzu wird im Feld PROGN das betreffende Programm angegeben.

```
DATA DIVISION

PROGN               8.      * PROGRAM NAME

PROCEDURE DIVISION

progn = 'CPGHLIH '.         * hl1 table h
prog cicsnco qpg.           * Newcopy of resident cics program
```

CPGCUR Setzen des Cursors auf den Bildschirm

Hierzu wird im Feld CPGCUR eine beliebige Zeile und Spalte angegeben.

```
DATA DIVISION

CPGCUR              4

PROCEDURE DIVISION

CPGCUR = '2412'.            * Zeile 24 Spalte 12
PROG CPGCUR QPG.            * Setze Cursor auf den Bildschirm
```

HELPA1 Aufruf der Masken bezogenen Hilfe mit PA1

HELPF1 Aufruf der Masken bezogenen Hilfe mit PF1

Die Hilfe kann extern in der QSF-Maske mit dem DEFine-Command aktiviert werden. Die Beschreibung ist unter dem Maskennamen in der Help-Library zu hinterlegen. Es kann für jedes Feld eine eigene Section mit Angabe des Feldnamens beschrieben werden.

Beim Define einer Bildschirmmaske (QSF Befehl DEF) kann ein QPG-Programm angegeben werden, z.B. HELPF1 in der QPG-Library:

```
   QQQQQ                                    V.L  UID  TERM  tt.mm.jj  ss.mmUHR
 QQ     QQ       Q üry
QQ       QQ       U ser
QQ       QQ         I nformation
QQ    QQ QQ         C ontrol
 QQ    QQQ            K it
   QQQQQ QQ                                                Quick Screen Facility
--------------------------------------------------------------------------------

QSF System Variable für Bild: PRKDMAP1

In diesem Bild können die Standardwerte für das oben genannte Bild
gesetzt werden. Ab der Eingabe werden diese Werte bei der Pflege des
Bildes anstatt Blank gesetzt.


Konstante Attribut   _  Farbe   T    E-H Wert _
Variable  Attribut   _  Farbe   R    E-H Wert _
Ausführung Auxilary _
QPG-Programm         HELPF1__   QPG_
Hintergrund Bilder   ________   ________   ________

--------------------------------------------------------------------------------
PF1 ==> Hilfe                                                PF12 ==> Ende
```

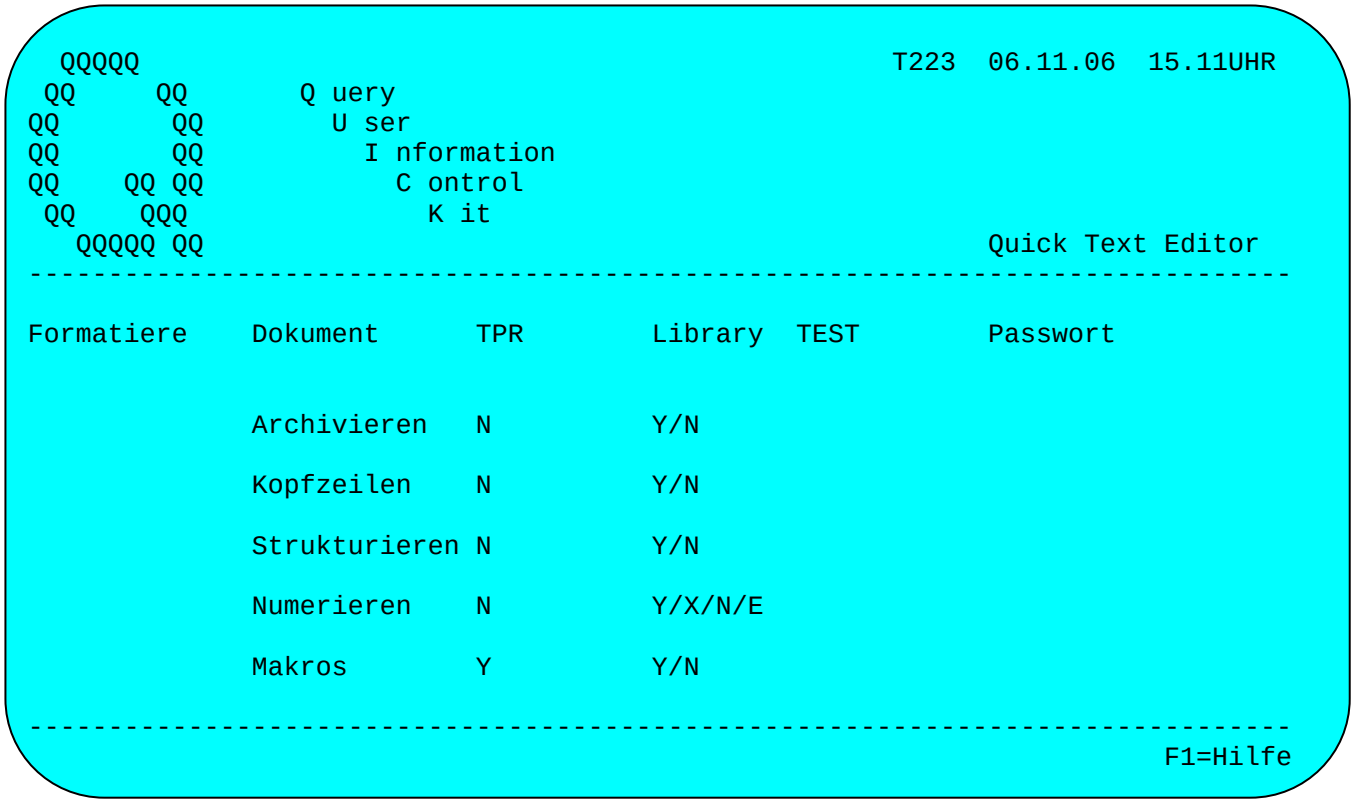
HELPA1 und HELPF1 zeigen die zu der Maske gespeicherte Hilfe (siehe QTF-Handbuch, Abschnitt HELP Facility) auf dem Bildschirm an. Dabei wird HELPA1 bei der Taste 'PA1' und HELPF1 bei der Taste 'PF1' aktiv.

Beispiel für das Hilfedokument, das mit HELPA1/HELPF1 aufgerufen wird:

```
§windows 1402-2137
§defatrb " pink rev
§defatrb ' yellow rev
§defatrb / tur rev
§section kdnra
'
' " Eingabe der Kunden-Nummer:    '
' "                               '
' " 00001 - 02999 für Kunden      '
' " 03000 - 09999 Interessenten   '
'                                 '
' / Test prkdmap1(help)           '
'                                 '
§section ....
.
.
.
```

PREP Preprocessor für QPG-Sourcecode

Nach Aufruf des Programms PREP erscheint folgendes Bild:



Da das Dokument verändert wird, empfiehlt es sich, dieses vorher zu archivieren. Die Beschreibung kann als Kopfzeilen in das Dokument kopiert werden. Der Programmcode kann strukturiert werden, dabei werden z.B. DO- und IF-Blöcke entsprechend eingerückt (Pretty Printer). Die Spalten 82 - 85 im Dokument können nummeriert werden, dabei werden mit Y alle Zeilen und mit X nur die Zeilen der Procedure Division nummeriert. E entfernt eine Nummerierung und N lässt die Spalten 82 - 85 unverändert. Werden Makros aktiviert, dann wird der normale Preprocessor aufgerufen. Siehe auch PREPO.

PREPO Preprocessor für QPG Sourcecode

Der Preprocessor wird normalerweise aus dem QTF-Menü aufgerufen. Er kann jedoch auch aus einem eigenen Programm im CICS ausgeführt werden. Als Parameter werden Dokumentname, Library und gegebenenfalls Passwort angegeben:

```
DATA DIVISION

DOKUM               8.        * Dokument Name
LIBR                4.        * Library
PASSW               8.        * Passwort

PROCEDURE DIVISION

DOKUM = 'INTADR  '.           * Dokument INTADR
LIBR  = 'PROG'.               * in Library PROG
PASSW = 'PR      '.           * mit Passwort PR
PROGRAM PREPO QPG.            * soll preprocessed werden.
```

Die Funktionen sind im Abschnitt Preprocessor Seite 1900 beschrieben.

QPCF Dataset für komprimierte Ausgabe von QPCF-Daten

Das Dataset QPCF ermöglicht die komprimierte Ausgabe von Daten, z.B. für Tabellen, Textareas und Listboxen im CPG5 oder CPGXML. Im Data Dictionary wird die Datei QPCF definiert:

```
Dateiname ..... QPCF     SP     Ein-/Ausgabe Art ...... U
Satzformat ..... F              Blocklaenge ...........
Satzlaenge ..... 00256          Schluessellaenge ...... 000
Dateiart ..... S                Dateiorganisation ..... P
Schluesselposition ....         Einheit / Library ..... QPG
```

Beispiel:

```
options  dat html
         file qpcf type sp.
         file cpgkdn
-d
         recnr               7 0.    * record number
-i
         file cpgkdn dd
-c
         open qpcf.
         write qpcf header.
         do 10.
            read cpgkdn
            if cpgfrc = 'EF'
               break
            endif
            write qpcf detail.
         enddo
         write qpcf trailr.
         close qpcf.
 -o
         file qpcf type header
                   'table c1;c2;c3;c4'
         file qpcf type detail sep ';'
                   kdnr
                   firma
                   plz + ' ' + ort
                   str1
         file qpcf type trailr
                   '$$CPGEND$$'
```

Folgende Operationen sind unterstützt: Für die Ausgabe OPEN oder DELET, WRITE und CLOSE und für die Eingabe SETLL, READ und RNDOM, wobei hier im Feld RECNR eine Satznummer vorgegeben werden kann.

QPCFXL wie QPCF, aber mit 999 Bytes Satzlänge.

Die Programmierung ist die gleiche wie bei QPCF, aber die Satzlange ist 999 Bytes.

QPGA Aufruf der QPG-Services.

Command, Library und Programmname und Return-Trans-Id werden vorgegeben.

```
DATA DIVISION

PRCMD               4.        * Program command
PROGN               8.        * Program name
PRLIB               4.        * Program library
PTRID               4.        * Return trans-id

PROCEDURE DIVISION

PRCMD = 'STA '.               * Display Status
PROGN = '        '.           * for all programs
PRLIB = 'PROG'.               * in library PROG
PTRID = 'MENU'.               * and return to user
PROGRAM QPGA QPG.             * call qpg services
```

QPGCHECK Check QPG Object Code

Bei variablen Programmaufrufen kann es sinnvoll sein, vor Aufruf erst zu prüfen, ob das angeforderte Programm auch verfügbar ist.Die Prüfung kann mit dem Baustein QPGCHECK erfolgen. An QPGCHECK werden Programmname und -library übergeben,und nach Rückkehr kann der Errorcode abgefragt werden und eine Statusmeldung im Feld INFO benutzt werden.

```
DATA DIVISION

EC                   3 0.     * error code
INFO                79.       * info message
PRLIB                4.       * program library
PROGN                8.       * program name

PROCEDURE DIVISION

PROGN = 'INTADR  '.           * program INTADR
PRLIB = 'PROG'.               * in library PROG
PROG QPGCHECK QPG.            * own error control
IF EC = 0.                    * ok
   .
   .                          * call Program
   .
ELSE.                         * error found
   DSPLY INFO.                * display message
   .
   .                          * error handling
   .
END
```

Der Errorcode EC hat folgende Bedeutung:

EC = 0 Programm ist umgewandelt und ausführbar.

EC = -1 Programm ist nicht ausführbar. Programmname fehlt oder bei der Übersetzung des Programms sind Fehler aufgetreten.

EC > 0 Programm konnte nicht übersetzt werden. Der Grund ist im Feld INFO beschrieben.

QPGDC QTF-Dokumentenverzeichnis

Die in einer QTF-Library gespeicherten Dokumente werden in eine Temporary Storage Queue ausgegeben. Diese Queue kann mit einem Folgeprogramm weiter verarbeitet werden. Der Queue-Name ist TermQPGV, wobei Term die Terminal-Id ist.

```
DATA DIVISION

LIBR                4.      * Library wahlw., Std. ist allg. Library
FRDOK               8.      * Ab  Dokument wahlw., Std. ist blank
TODOK               8.      * Bis Dokument wahlw., Std ist 9999

PROCEDURE DIVISION (Beispiel)

LIBR  = 'PROG'.             * Library PROG speichern
PROGRAM QPGDC    QPG.       * Verzeichnis auf TS QPGV speichern.
EXITI 'QSTS'.               * Kontrolle mit QSTS
```

Der Storage QPGV wird wie folgt erstellt:

Stelle	1 - 8	Dokumentname
	9 - 48	Beschreibung
	49 - 52	Erstellungsdatum (gepackt 6,0)
	53 - 55	Erstellt User-ID
	56 - 59	Änderungsdatum (gepackt 6,0)
	60 - 62	Geändert User-ID

QPGLOG Druckt das Debug Protokoll aus.

Die Drucker-Id kann angegeben werden. Der Logfile (TS-Queue QPGL) wird nach dem Drucken gelöscht.

```
DATA DIVISION

CPGDID              4.        * Drucker-Id

PROCEDURE DIVISION

CPGDID = 'PRT0'.              * Use Batch Printer SYSLST
PROGRAM QPGLOG QPG.           * And Print Logfile
```

Dieses Programm kann benutzt werden, um den mit DEBUG erstellten Logbereich zu drucken, z.B. bei Batch-Ausführung mit QPGUTIL, wobei ein Debug Code angegeben wurde. Da der Logfile nach dem Drucken gelöscht wird, so kann QPGLOG bei umfangreichen Tests auch mehrfach (z.B. in einer Schleife) aufgerufen werden.

Beim Debuggen wird der Logbereich nach 10.000 Einträgen automatisch gerollt, damit kein Überlauf eintritt und immer die letzten Einträge ausgewertet werden können.

QPGNCOPY QPG Newcopy

Von einem bestimmten QPG-Programm soll ein New-Copy ausgeführt werden.

```
DATA DIVISION

PROGN               8.        * Programmname erforderlich
PRLIB               4.        * Programm Library erforderlich
INFO               79.        * Return Information

PROCEDURE DIVISION

PRLIB  = 'TEST'.              * In der Library TEST
PROGN  = 'TEST    '.          * ist für Dokument TEST
PROGRAM QPGNCOPY QPG.         * ein New-Copy auszuführen
EXITI 'QPG'.                  * und mit QPG zu kontrollieren
```

Bei der Library LIST wird ein Newcopy vom LIST-Dokument ausgeführt. Die Library JOB kann ebenfalls zum Laden neuer Power Jobs benutzt werden.

QPGPREP Compile QPG-Programm.

Ein QPG-Programm soll übersetzt werden. Hierbei erfolgt gleichzeitig eine Diagnostik (Syntaxprüfung). Wird bei ERRCTL nichts (' ') angegeben so wird die Generierung mit einer Fehlermeldung abgebrochen. Wird ERRCTL = 'X' angegeben, so werden bei einem Fehler die Felder EC, RC, EI und INFO (Klartext) gefüllt und das Programm kann seine eigene Fehlerbehandlung durchführen. Falls der Objectcode zu dem Programm bereits existiert, so wird hierfür automatisch ein New-Copy ausgeführt.

```
DATA DIVISION

EC                   3 0.    * error code
EI                   1.      * error indicator
ERRCTL               1.      * error control X=exit
INFO                79.      * info message
PRLIB                4.      * program library
PROGN                8.      * program name
RC                   3 0.    * return code

PROCEDURE DIVISION

PROGN = 'INTADR  '.          * program INTADR
PRLIB = 'PROG'.              * in library PROG
ERRCTL = 'X'.                * own error control
PROG QPGPREP QPG.            * prepare program
IF EI >< ' ' OR.
   EC >< 0   OR.
   RC >< 0.                  * Error found
   DSPLY INFO.               * Display Error Message
   .
   .                         * Error Handling
   .
END
```

QPGSTRT Start weiteres QPG-Programm

Es wird ein weiteres QPG-Programm z.B. an einem anderen Terminal oder als Hintergrundprogramm (NONE-Terminal Task) im CICS gestartet. Dabei können Daten in der Common Area übergeben werden. Hiermit können sehr einfach Druckprogramme auf einem Drucker gestartet werden. Diese erlauben dann z.B. das direkte Drucken ohne Umweg über Transient Data.

```
DATA DIVISION

PRLIB               4.      * start program library
PROGN               8.      * start program
STERM               4.      * start terminal id
STIME               7 0.    * start time interval
STYPE               1.      * start type: T = time (hhmmss)

PROCEDURE DIVISION

edit cpgcom .               * set data into comarea
progn = 'PRTEL   '.         * program is prtel
prlib = 'PROG'.             * in library prog
sterm = 'L86C'.             * at printer l86c
prog qpgstrt qpg.           * start it

OUTPUT DIVISION

field cpgcom
                5 '00001'. * from key
               10 '02999'. * to key
```

Mit den Variablen STIME und STYPE kann die Verarbeitung zu einer festen Uhrzeit oder nach einem bestimmten Zeitintervall gestartet werden, z.B.:

```
stime = 170000.             * start zeit ist 17:00:00 Uhr
stype = 'T'.                * start type ist T = Zeit (hhmmss)

stime = 20.                 * start in 20 sekunden
stype = ' '.                * start type ist intervall (ss, default)
```

Mit Angabe 'NONE' bei STERM kann das Programm in einer Hintergundtransaktion gestartet werden.

```
sterm = 'NONE'.             * start als NONE-Terminal Task
```

Mit dem Aufruf von QPGSTRT in der Servicelibrary QPG wird eine weitere CICS-Task (auf dem Drucker L86C) gestartet.

Das aufgerufene QPG-Programm erzeugt mit den übergebenen Daten aus der Common-Area eine Liste auf dem Drucker (L86C). Der List-Befehl wurde ohne Angabe der Drucker-ID codiert. Hiermit werden die Daten direkt auf dem Drucker ohne Zwischenspeichern auf Transient Data ausgegeben:

```
*        telefonliste drucken                              PROG.PRTEL

options  dat.
         file cpgkdn.                * kundenstamm
-d
         page                 3 0.   * page counter
         cpglct               3 0.   * line counter
-i
         field cpgcom
                        1    5   kdnra
                        6   10   kdnrb
         file cpgkdn dd
-c
         select cpgcom
         list prtel header
         kdnra setll cpgkdn
         do loop

            read cpgkdn.             * get kunde
            if cpgfrc = 'EF' or
               kdnra > kdnrb.        * kdnra is current key
               break
            endif
            list prtel detail
         enddo
         rndom cpgkdn
         list prtel trailr
```

LIST-Dokument PRTEL zum Druckprogramm:

```
§section header
*************************************************************************
*
*        Telefonliste                  vom: §udate         Seite  §page:ps
*
*  Tel.-Nr.                   Firma                Plz / Ort
*************************************************************************
§section detail    §if cpglct > 60 *   bei overflow neue seite anfangen
§perform trailr
§perform header
§section detail
*  §telnr                     §Firma             #§plz  §ort
§section trailr
*
*************************************************************************
§newpage
```

QTFA Aufruf des Textsystems.

Auswahl, Dokument, Library usw. werden übergeben. Es sind alle Datenfelder angegeben, allerdings müssen nur die tatsächlich benötigten Felder definiert werden. Die wichtigsten Felder sind mit Großbuchstaben gekennzeichnet:

```
DATA DIVISION

arcdtm              7.        * archiv date/time
arcdok              1.        * archivieren dokument
archiv              1.        * auswahl x=archiv
AUSW                1.        * FUNKTIONSAUSWAHL
ausw2               1.        * auswahl 2
ausw3               1.        * auswahl 3
chr          10 *   1.        * sonderzeichen
chx          10 *   1.        * hex übersetzung
colore              2.        * eh und farbe fehlermeldungen
colors              1.        * user defined colors y/n
color1              1.        * farbe input text
color2              1.        * farbe display text
color3              1.        * farbe markierter text
color4              1.        * farbe protected text
cpghic              4.        * hl1 interface control
ctlchr              1.        * druckersteuerzeichen
dclass              1.        * drucker klasse
DESCR              40.        * BESCHREIBUNG
dkctl               1.        * dokument control character
dkslg               3 0.      * dokument seitenlänge
dktyp               1.        * Allg Baust Help List Prog Test
dkzlg               3 0.      * dokument zeilenlänge
DOKUM               8.        * DOKUMENTNAME
DOKUN               8.        * NEUER DOKUMENTNAME
DRID                4.        * DRUCKER ID
EC                  1.        * ERROR CODE
ERRC                1.        * ERROR CONTROL
filen2              8.        * neue datei bei Copy
FUNC                1.        * O=OPEN X=EXECUTE Z=CHECK
hex          10 *   2.        * hexadezimalwerte
info               79.        * nachrichten und fehlermeldungen
kopien              3 0.      * zusätzliche kopien
LIBR                4.        * DOKUMENT LIBRARY
LIBN                4.        * DOKUMENT LIBRARY NEU.
nulls               1.        * nulls
PASSW               8.        * PASSWORT
passwn              8.        * neues passwort
prlib               4.        * program library
progn               8.        * program name
randb               1.        * randbegrenzung rechts
randl               3 0.      * rand links
randr               3 0.      * rand rechts
randv               3 0.      * randverschiebung links
RC                  3 0.      * RETURNCODE
```

```
sblib              4.        * baustein library
seitea             4.        * seite ab
seiteb             4.        * seite bis
sessnr             1 0.      * session nr.
spqjm              1.        * sperre qjm
spqpg              1.        * sperre qpg
spqtm              1.        * sperre qtm
stlib              4.        * start library
stwort            18.        * stichwort
tab          20 *   3 0.      * tabulatoren
tabchr             1.        * tabulator zeichen
tasize             1.        * alternate screen size used
trid               4.        * return trans-id
uctr               1.        * bildschirmstatus
uctrd              1.        * uctran switch for printer
uid                3.        * benutzer id
usratr             4.        * user attribub
usrdef            10.        * user definitionen
vdatum             6 0.      * verfalldatum
verzab             4.        * verzeichnis ab
xedstr            32.        * edit string
xpostr            32.        * positionieren string
xprot              1.        * protection code
xscstr            32.        * scan string

PROCEDURE DIVISION

func = 'O'.                  * open
prog qtfa qpg.               * get user default valüs
ausw = 'E'.                  * edit
dokum = 'MUSTER  '.          * dokument MUSTER
libr = 'TEST'.               * in library TEST
func = 'X'.                  * execute
trid = 'USER'.               * return to user
prog qtfa qpg.               * call qtf
```

QTFC Textdataset

Dokumente können mit OPEN, CLOSE, READ, WRITE, UPDATE usw. verarbeitet werden. Im Feld RC wird der Returncode übergeben. Diese Schnittstelle entspricht dem HL1-Dataset HQTFC, das im QTF-Handbuch auf Seite 7200 be-
schrieben ist.

Auswahl, Dokument, Library usw. werden übergeben. Es sind alle Datenfelder angegeben, allerdings müssen nur die tatsächlich benötigten Felder definiert werden. Die wichtigsten Felder sind mit Großbuchstaben gekennzeichnet:

```
DATA DIVISION

bseite              4 0.      * bis seite
descr              40.        * beschreibung des dokuments
dkctl               1.        * druckersteuerzeichen
dkelg               3 0.      * effektive länge bei read od.
dkslg               3 0.      * seitenlänge des dokuments
dkzlg               3 0.      * zeilenlänge des dokuments
DOKUM               8.        * DOCUMENT NAME
dokun               8.        * neuer dokumentname
dop1                1.        * zeilenbreite begrenzen
info               79.        * informationszeile
libn                4.        * library neu
LIBR                4.        * PRIVATE LIBRARY
OC                  2.        * OPERATION CODE
PASSW               8.        * PASSWORD
passwn              8.        * passwort neu
RC                  3 0.      * RETURNCODE
SATZ              256.        * EIN-AUSGABESATZ
SEITE               4 0.      * PAGE
uid                 3.        * benutzerkurzzeichen
vdatum              6 0.      * verfalldatum
xprot               1.        * protectioncode
ZEILE               4 0.      * LINE

PROCEDURE DIVISION

oc = 'OI'.                    * open for input
DOKUM = 'MUSTER  '.           * Document is MUSTER
LIBR = 'TEST'.                * in library TEST
prog qtfc qpg.                * call textdataset
if rc = 0.                    * is open ok
   do 10
      oc = 'R '.              * read
      prog qtfc qpg.          * call textdataset
      if rc = 2.              * end of page
         zeile = 0.           * first line
         seite = seite + 1.   * in next page
         cont.                * continue
      end
      .
      .                       * process satz
      .
   end
   oc = 'C '.                 * close document
   prog qtfc qpg.             * call textdataset
end
```

QTFDV Drucken Verzeichnis

Hiermit wird das Verzeichnis einer Library gedruckt. Library, Druckername und gegebenenfalls Von- und Bis-Dokument werden übergeben.

```
DATA DIVISION

cpgdid              4.        * printer id
frdok               8.        * from document
libr                4.        * library
todok               8.        * to document

PROCEDURE DIVISION

cpgdid = 'QTFS'.              * print on Storage QTFS
frdok = 'HUGO    '            * from document HUGO
todok = 'OTTO'.               * to document OTTO
libr = 'TEST'.                * in library TEST
prog QTFDV QPG.               * print view
```

QTFK Kopieren in QTF-Dokument

TS-Name, Dokument, Library usw. werden übergeben. Es sind alle Datenfelder angegeben, allerdings müssen nur die tatsächlich benötigten Felder definiert werden. Die wichtigsten Felder sind mit Großbuchstaben gekennzeichnet:

```
DATA DIVISION

DESC               40.        * DESCRIPTION
DKSLG               3 0.      * DOKUMENT PAGE LENGTH
DKZLG               3 0.      * DOKUMENT LINE LENGTH
DOKUM               8.        * DOCUMENT NAME
drid                4.        * input from transient data
filenm              8.        * filename
info               79.        * messages
LIBR                4.        * LIBRARY
passw               8.        * password
REP                 1.        * REPLACE C=CHECK X=YES
TSNAME              4.        * INPUT FROM TS NAME
tstid               4.        * input from ts teminal-id
vdatum              6 0.      * expiration date
xprot               1.        * protection code

PROCEDURE DIVISION

desc = 'Created by qpg qtfk'. * set description
dkslg = 72.                   * set page length
dkzlg = 72.                   * set line length
dokum = 'MUSTER  '.           * document is MUSTER
libr = 'TEST'.                * in library TEST
rep = 'X'.                    * replace if it exists
tsname = 'QTFS'.              * storage is QTFS
prog qtfk qpg.                * copy ts to document
```

Über die Ausführung erfolgt ein Hinweis im Klartext im Feld INFO. Diese Schnittstelle entspricht dem HL1-Modul HQTFK, das im QTF-Handbuch auf Seite 7650 beschrieben ist.

QTFLIST QTF-Dokument auflisten

Ein QTF-Dokument soll aufgelistet werden.

```
DATA DIVISION

DOKUM               8.      * Dokumentname ist erforderlich
DRID                4.      * Drucker-Id
PASSW               8.      * Password wahlweise
LIBR                4.      * Library wahlweise
RC                  2 0.    * Returncode
SEITEA              5 0.    * Seite ab
SEITEB              5 0.    * Seite bis

PROCEDURE DIVISION

DOKUM = 'TPR'.              * Dokument TPR
DRID  = 'INFO'.             * Drucken auf Drucker INFO
PROGRAM QTFLIST  QPG.       * Aufruf List Programm
IF RC >< 0
   :                        * Fehler
END
```

Der Returncode RC gibt folgende Fehler an (wie bei Dataset HQTFC):

```
01    Dokument ist nicht verfügbar.
02    Ende der Seite.
03    Ende des Dokuments.
04    Dokument ist geschützt.
05    Dokument ist in Arbeit an Bildschirm XXXX
06    Kein weiterer Platz für das Dokument vorhanden.
07    Dokumentname vorhanden.                          (beim Anlegen)
08    Dokument nicht vorhanden.
10    Ungültige Seitenangabe.
12    Angegebene Seiten kopiert.                       (beim Kopieren)
13    Gleiche Namen sind nicht zulässig.               (beim Kopieren)
14    Angegebene Seiten kopiert. Neues Dokument angelegt.  (Kopieren)
16    Seite(n) nicht vorhanden.
17    Dokument gelöscht.                               (beim Löschen)
18    Seiten gelöscht.                                 (beim Löschen)
19    Library nicht verfügbar.
20    Library nicht vorhanden.
22    Der Dokumentname ist ungültig.
32    Zieldokument ist geschützt.
```

QTFLOAD QTF-Dokument laden

Ein QTF-Dokument soll aus QPCF-Daten geladen werden.

```
DATA DIVISION

DOKUM               8.              * document
LIBR                4.              * library
PASSW               8.              * password
DESCR              40.              * description for create document
DKSLG               3 0.            * page length for create document
DKZLG               3 0.            * line length for create document
RC                  2 0.            * returncode
INFO               79.              * messages
XPROT               1.              * protectioncode for create docum

PROCEDURE DIVISION

dokum = 'TPR     '.                 * Dokument TPR
libr  = '    '.                     * Allgemeine Library
PROGRAM QTFLOAD QPG.                * Aufruf Load-Programm
IF RC >< 0
   :                                * Fehler
END
```

Der Dokumentname muss angegeben sein. Ist das Dokument nicht vorhanden, so wird es angelegt. Beim Anlegen sollten die Felder DESCR, DKSLG und DKZLG gefüllt sein. Bei geschützten Dokumenten wird das Feld PASSW benötigt und gegebenenfalls beim Anlegen XPROT.

Wenn ein Fehler aufgetreten ist,kann er mit dem Returncode RC analysiert werden. Im Feld INFO ist die Fehlermeldung im Klartext enthalten.

QTFLV Library Verfalldatum

Mit diesem Serviceprogramm kann das Verfalldatum für alle oder für eine Gruppe von Dokumenten in einer Library gesetzt werden. Die Library muss verfügbar sein, dh. mit '*OA' oder '*BA' (CICS bzw. Batch) gekennzeichnet sein.

```
DATA DIVISION

LIBARC              1.        * ' '=Normal, 'A'=Archiv
LIBR                4.        * library ist erforderlich
FRDOK               8.        * from document
TODOK               8.        * to document
VDATUM              6 0.      * Verfalldatum (ttmmjj) ist erforderlich
RC                  3 0.      * return code

PROCEDURE DIVISION

LIBR  = 'TEST'.               * In der Library TEST
FRDOK = 'TPR'.                * soll ab Dokument TPR
MOVEN UDATE TO VDATUM.        * das Tagesdatum als
PROGRAM QTFLV  QPG.           * Verfalldatum gesetzt werden
IF RC >< 0
   :                          * Fehler
END
```

Folgende Returncodes sind möglich:

```
RC = 0.                       * Alles ok
RC = 1.                       * Library nicht gefunden
RC = 2.                       * Library nicht verfügbar
RC = 4.                       * keine Dokumente gefunden
```

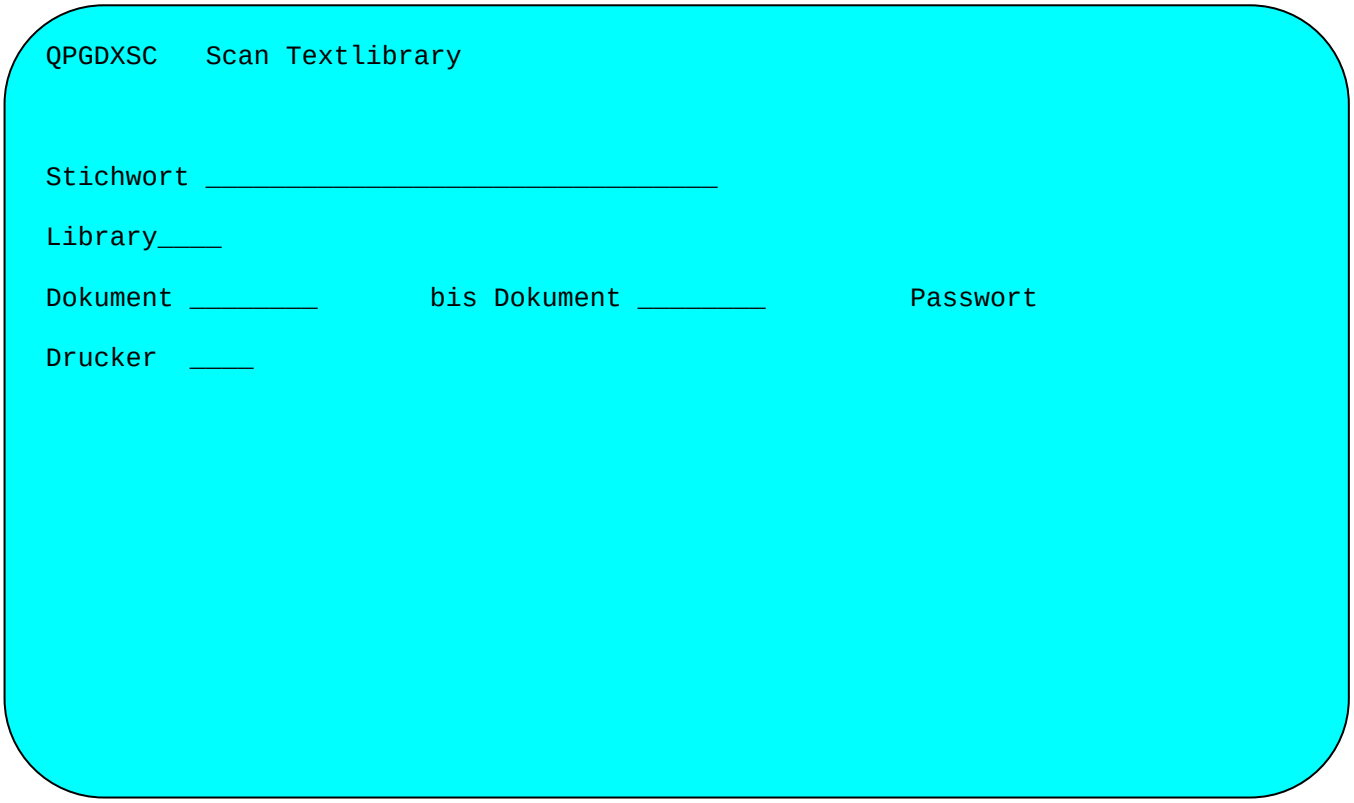
QTFSCAN Durchsuchen einer Textlibrary nach einem Stichwort

Stichwort, Library, Drucker-Id und eventuell Von- und Bis-Dokument werden am Bildschirm eingegeben. Der Inhalt der verfügbaren und ausgewählten Dokumente in der Library wird nach dem Stichwort überprüft. Auf dem Drucker wird eine Crossreference der Dokumente ausgedruckt, in denen das Stichwort vorkommt. Das Stichwort wird in Großbuchstaben übersetzt. Der Text wird intern zum Durchsuchen ebenfalls übersetzt, so dass das Stichwort auch dann gefunden wird, wenn es klein geschrieben wurde.

```
PROCEDURE DIVISION

PROGRAM QTFSCAN QPG.          * Durchsuchen nach Stichwort
```

Die Parameter werden am Bildschirm angefordert:



QTFSCANS Durchsuchen einer Textlibrary nach einem Stichwort (Subroutine)

Stichwort, Library, Drucker-Id und eventuell Von- und Bis- Dokument werdenvorgegeben. Ansonsten ist die Funktion identisch mit QTFSCAN.

QTFSCANS wird als Unterprogramm aufgerufen und läuft automatisch ab.

```
DATA DIVISION

cpgdid              4.        * printer id
txdokf              8.        * from document
txdokt              8.        * to document
txlib               4.        * library
txpwd               8.        * password
txsstr             32.        * scan string

PROCEDURE DIVISION

txsstr = 'FILE'.              * search for FILE
txlib = 'PROG'.               * in library PROG
cpgdid = 'L86C'.              * and print to L86C
prog qtfscans qpg.            * the list of documents
```

QTFSORT QTF-Dokument sortieren

Ein QTF-Dokument soll sortiert werden.

```
DATA DIVISION

DOKUM               8.      * Dokumentname ist erforderlich
PASSW               8.      * Password wahlweise
LIBR                4.      * Library wahlweise
SEITE               4 0.    * Page wahlweise, Standard ist Seite 1
RC                  2 0.    * Returncode (sh. Seite 5535)
TRAN                1.      * Translate 1,2,3

PROCEDURE DIVISION

DOKUM = 'TPR'.              * Dokument TPR
PROGRAM QTFSORT  QPG.       * Aufruf Sort-Programm
IF RC >< 0
   :                        * Fehler
END
```

Es wird eine Map eingeblendet, um die Sortierbegriffe einzugeben:

```
QPGDXS    Sortieren QTF-Dokument


Dokument                Passwort

          Seite       1

Library

Übersetz.               (1=Groß  2=äöüÄÖÜ 3=beides)

Sortieren     von - bis   art(a/d)
              ---   ---     -
          1.    1    10     A
          2.
          3.
```

Beim Sortieren von QTF-Dokumenten werden auch Kleinbuchstaben und Sonderzeichen berücksichtigt, indem bei Übersetzen ein Eintrag erfolgt:

'1' Übersetzt den Sortierbegriff in Großbuchstaben, damit werden Groß- und Kleinbuchstaben gemischt, z.B. in der Folge A,a,b,B...
'2' Übersetzt Sonderzeichen ä,ö,ü,Ä,Ö,Ü , um diese zu den 'normalen' Buchstaben zu mischen, in der Folge z.B. a,ä,ö,o,u,ü,Ä,A,O,Ö,Ü,U
'3' Kombiniert die Übersetzungen '1' und '2', damit erscheinen die Buchstaben in der Folge z.B. a,ä,A,Ä, ... ö,O,o,Ö, ... Ü,u,ü,U

Es erfolgt eine Syntaxprüfung der Sortierparameter.

QTFSORTS QTF-Dokument sortieren (Subroutine)

Ein QTF-Dokument wird nach den angegebenen Kriterien sortiert. Es wird keine Map eingeblendet, um die Sortierbegriffe einzugeben, daher müssen die Sortierparameter korrekt vorgegeben werden.

```
DATA DIVISION

DOKUM               8.      * Dokumentname ist erforderlich
PASSW               8.      * Password wahlweise
LIBR                4.      * Library wahlweise

SEITE               4 0.    * Page wahlweise, Standard ist Seite 1
RC                  2 0.    * Returncode (sh. Seite 5535)
S                   3 0.    * Anzahl Sortparameter erforderlich 1-3
PARM      3    *    8.      * Sortparameter
TRAN                1.      * Translate 1,2,3

PROCEDURE DIVISION

DOKUM = 'TPR'.              * Dokument TPR
S = 1.                      * 1 Sortierbegriff
PARM(1) = '000110CA'.       * von St. 1, 10 Byte, Character, Aufstg.
PROGRAM QTFSORTS QPG.       * Aufruf Sort Unterprogramm
IF RC >< 0
   :                        * Fehler
END
```

Der Sortparameter PARM ist folgendermaßen belegt:

```
Stelle  1 - 4  von Position
        5 - 6  Länge
            7  C für Character, P für gepacktes Format
            8  A für aufsteigend, D für absteigend
```

Die Übersetzung (sh. QTFSORT) kann im Feld TRAN angegeben werden:

```
'1'  Übersetzt den Sortierbegriff in  Großbuchstaben
'2'  Übersetzt Sonderzeichen ä,ö,ü,Ä,Ö,Ü
'3'  Kombiniert die Übersetzungen '1' und '2'
```

Es erfolgt eine Syntaxprüfung der Sortierparameter. Bei einem Fehler wird im Feld RC der Returncode 2 übergeben.

QUERY Aufruf des Report-Generators

Der Programmname wird vorgegeben

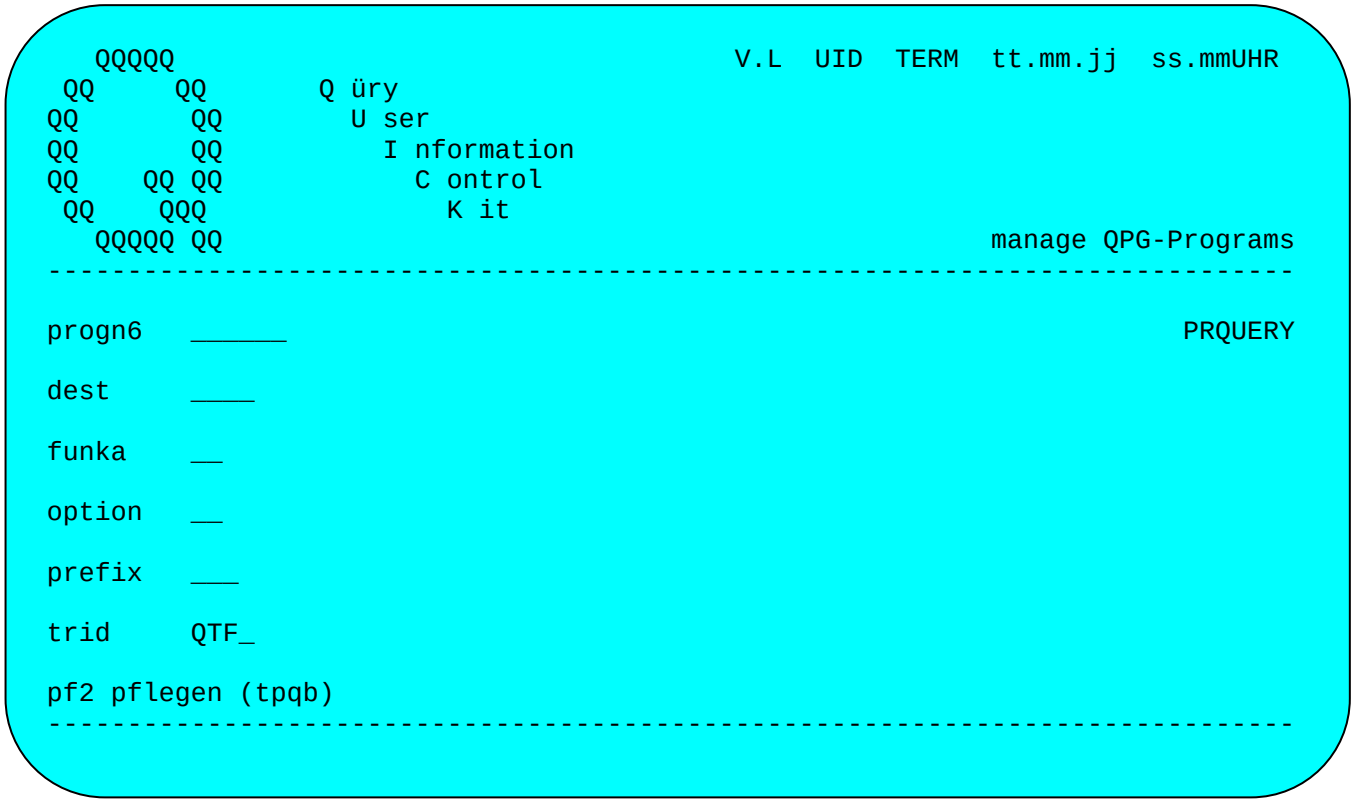
```
DATA DIVISION

PROGN6              6.        * PROGRAMMNAME
DEST                4.        * DRUCKERNAME
TRID                4.        * RETURN TRANS-ID
FUNKA               2.        * ANZ='  ', DRUCKEN='01', BEIDES='02'
OPTION             60.        * TEMP. AENDERUNGEN DER H-KARTE
PREFIX              3.        * Mandant
func                1.        * P=pflegen
mapnam              8.

PROCEDURE DIVISION

twald.                       * load twa
if cpgfrc = '  '.            * twa found
   map prquery.              * get fields from map
   if mapnam = 'PRQUERY'.    * is it the right map
      if cpgmpf = 'P3'.      * check pf3
         prog menu.          * if so return to menu
      end.                   *
      prog query qpg.        * save Query parameters
      if cpgmpf = 'P2'.      * check pf2
         exiti 'TPQB'.       * if so call maintenance
      else.                  * else
         exiti 'TPQO'.       * execute Query online
      end.                   *
   end.                      *
end.                         *
trid = 'QTF'.                * return to qtf
mapo prQuery.                * put map
twasv.                       * save twa
task.                        * start this program in next task
```

In dem Beispiel werden in der Map PRQUERY die Parameter für den Query-Aufruf eingegeben:



QXFA Aufruf des Expertensystems

Bis zu drei Stichworte können angegeben werden.

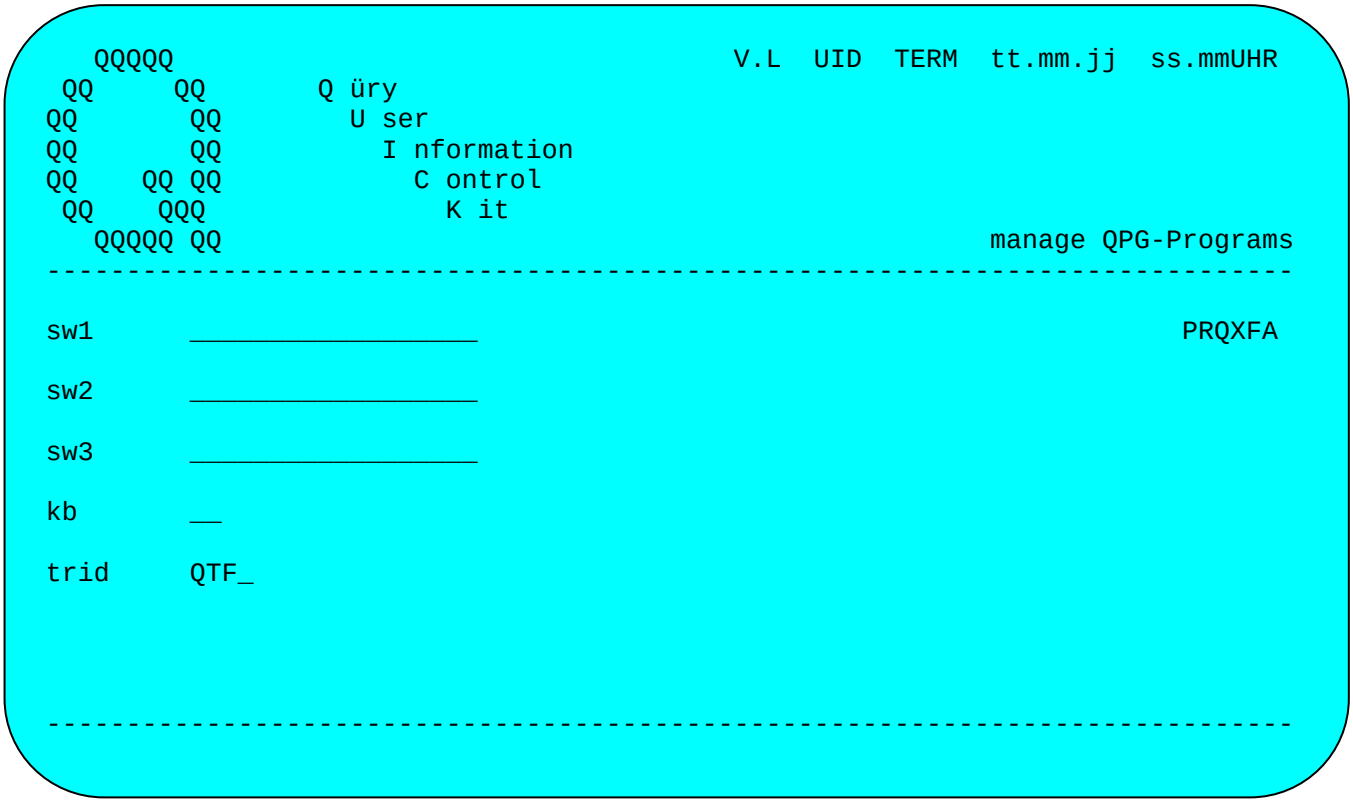
```
DATA DIVISION

kb                  2.        * wissensbank
sw1                18.        * stichwort 1
sw2                18.        * stichwort 2
sw3                18.        * stichwort 3
trid                4.        * return trans-id
mapnam              8.        * map mame

PROCEDURE DIVISION

twald.                        * load twa
if cpgfrc = '  '.              * twa found
   map prqxfa.                * get fields from map
   if mapnam = 'PRQXFA'.      * is it the right map
      if cpgmpf = 'P3'.       * check pf3
         prog menu.           * if so return to menu
      end.                    *
      prog qxfa qpg.          * call qxf
   end.                       *
end.                          *
trid = 'QTF'.                 * return to qtf
mapo prqxfa.                  * put map
twasv.                        * save twa
task.                         * start this program with next task
```

In dem Beispiel werden in der Map PRQXF die Parameter für den QXF-Aufruf eingegeben:



TASK Transaktionssteuerung

Das Programm TASK wird zur Ablaufsteuerung bei transaktionsorientierten QPG-Programmen benutzt. TASK dient zur Kommunikation zwischen aufrufendem und aufgerufenen Programm. Vor Aufruf der 'Folge'-Transaktion teilt das aufrufende Programm der TASK-Steuerung die Rückkehrinformationen mit (QPG-Programmname und Library oder CICS Trans-Id). Das aufgerufene Programm benutzt Programm TASK für den Rücksprung.

Mit 'S' im Feld PRGOPC erfolgt der Start einer QPG-Task, wobei angegeben werden kann, an welcher Stelle (QPG-Programmname und Library oder Return Trans-Id) der Ablauf bei Return fortgesetzt wird. Mit 'R' im Feld PRGOPC erfolgt der Rücksprung.

Das Feld PRGVAR enthält bei 'S' die Return-Informationen und zwar entweder von 1-8 den QPG-Programmnamen und von 9-12 die Library oder in Stelle 1-8 Blank und von 9-12 die CICS Return Trans-Id. Dieses Feld wird nur im aufrufenden Programm benötigt.

Der Ablauf kann bis zu 10 Stufen tief verschachtelt werden.

```
DATA DIVISION

prgopc              2.      * program operation code
prgvar             12.      * program name and library

PROCEDURE DIVISION (Beispiel nächste TASK im aufrufenden Programm)

prgopc = 'S'.               * start
prgvar = 'AKBM    TEST'.    * Program AKBM in Library TEST
prog task qpg.              * Save Task Informations
```

```
PROCEDURE DIVISION (Beispiel Return im aufgerufenen Programm)

prgopc = 'R'.               * return
prog task qpg.              * task
```

TXTA Aufruf des Textsystems. Wie QTFA, jedoch mit TX-Feldnamen.

Für den Aufruf des Textsystems steht außer QTFA noch eine weitere Schnittstelle zur Verfügung, in der aber die Feldnamen genormt sind und mit TX beginnen. Außerdem wird diese Schnittstelle vom Preprocessor PREP unterstützt. Mit folgenden Eingaben kann z.B. ein QPG-Programm erstellt werden, das den QTF-Editor aufruft,um ein Dokument zu editieren:

```
-d
§text    define
-c
§text    edit  testdok
```

Der Preprocessor generiert hieraus folgenden Programmcode:

```
-d
*text    define
         txdesc             40.      * text description
         txdid               4.      * text printer id
         txdoc               8.      * text documentname
         txdocn              8.      * text new documentname
         txfunc              1.      * text function O=open X=exit
         txinf              79.      * text information line
         txlib               4.      * text document library
         txlibn              4.      * text new dokument library
         txline              4 0.    * text current line

         txoc                2.      * text operation code
         txpage              4 0.    * text page
         txpagt              4 0.    * text page to
         txrc                3 0.    * text returncode
         txrec             256.      * text data record
         txtrid              4.      * text return trans-id
-c
*text    edit  testdok
         txfunc = 'O'.               * open
         prog txta qpg.              * qtf default valüs
         txoc = 'E'.                 * edit document
         txdoc = 'TESTDOK '.         * move document name
         txlib  = '    '.            * move document library
         txtrid = 'QPG'.             * return to qpg
         txfunc = 'X'.               * execute
         prog txta qpg.              * qtf services
```

TXTC Textdataset. Wie QTFC, jedoch mit TX-Feldnamen.

Für das Textdataset steht außer QTFC noch eine weitere Schnittstelle zur Verfügung, in der aber die Feldnamen genormt sind und mit TX beginnen. Außerdem wird diese Schnittstelle vom Preprocessor PREP unterstützt. Mit folgenden Eingaben kann z.B. ein QPG-Programm erstellt werden, das z.B. mit TXTC ein Dokument anlegt und mit Leerzeilen füllt:

```
-d
§text    define
-c
§text    create testdok # beispiel#dokument
§text    write anfang
         fill ' ' to txrec
         do 10.
§text       write
         end
§text    write ende
§text    close testdok
```

Der Preprocessor generiert hieraus folgenden Programmcode:

```
-d
*text    define
         txdesc             40.      * text description
         txdid               4.      * text printer id
         txdoc               8.      * text documentname
         txdocn              8.      * text new documentname
         txfunc              1.      * text function O=open X=exit
         txinf              79.      * text information line
         txlib               4.      * text document library
         txlibn              4.      * text new dokument library
         txline              4 0.    * text current line
         txoc                2.      * text operation code
         txpage              4 0.    * text page
         txpagt              4 0.    * text page to
         txrc                3 0.    * text returncode
         txrec             256.      * text data record
         txtrid              4.      * text return trans-id
-c
*text    create testdok # beispiel#dokument
         txoc  = 'ZA'.               * create document
         txdoc = 'TESTDOK '.         * move document name
         txdocn = txdoc.
         txlib  = '    '.            * move document library
         txlibn  = txlib.
         txdesc = 'beispiel dokument       '.     * move description
         program txtc qpg.           * call textdataset
*text    write anfang
         txrec  =  'anfang                  '
         txoc  = 'N '.               * write document
         program txtc qpg.           * call textdataset
         fill ' ' to txrec
         do 10.
*text       write
            txoc  = 'N '.            * write document
            program txtc qpg.        * call textdataset
         end
*text    write ende
         txrec  =  'ende                    '
         txoc  = 'N '.               * write document
         program txtc qpg.           * call textdataset
*text    close testdok
         txoc  = 'C '.               * close document
         program txtc qpg.           * call textdataset
```

TXTH Aufruf Help Facility mit TX-Feldnamen.

Die Hilfe kann mit folgenden Parametern aufgerufen werden:

```
DATA DIVISION

txdoc               8.      * text help document
txdid               4.      * text printer
txectl              1.      * text help error control
txhlpa              1.      * text help auswahl
txhlpo              1.      * text help option
txhlps              8.      * text help section
txlib               4.      * text help library
txpage              4 0.    * text help page
txrc                3 0.    * text help return code
txuid               3.      * text user-id

PROCEDURE DIVISION

txdoc = 'KUNDEN  '.         * help document is KUNDEN
txhlps = 'KDNRA '.          * section is KDNRA
prog txth qpg.              * display help informations
```

Diese Schnittstelle entspricht dem HL1-Modul HQTFK (siehe QTF-Handbuch).

Produktion

QPG eignet sich sowohl für die Entwurfs- und Testphase, als auch für die Produktionsphase.

Es empfiehlt sich, unterschiedliche QPG-Libraries für Test und Produktion zu benutzen und gegebenenfalls auch den Entwurf in einem TEST-CICS und die Ausführung in einem PROD-CICS auszuführen.

In der Produktions-Library sollten häufige Newcopy-Aufrufe vermieden werden.

Um die Ausführung im Produktionssystem schneller und unabhängig von QTF ausführen zu können, ist ein Compiler verfügbar, der QPG-Produktionslibraries als ausführbare Phasen erstellt.

Libraries

Es können standardmäßig bis zu je 1.000 Test- und Produktionslibraries eingerichtet werden. Die Libraries werden mit Ausnahme der standardmäßig vorhandenen Libraries PROG, QPG, TASK und TEST automatisch beim ersten Aufruf in der Directory verzeichnet. Eigene QPG-Libraries können hierdurch einfach angelegt werden. Die Pflege der Directory QPGPXD entfällt.

Compiler

QPG-Programme können umgewandelt und als Phase gelinkt werden. Hiermit ist eine saubere Trennung zwischen Test- und Produktionsumgebung gewährleistet. Zusätzlich entfällt die für den ersten Aufruf eines Programms erforderliche Generierung des Pseudocodes. Für die aufrufenden Programme ändert sich bei der Ausführung nichts, sie brauchen auch nicht neu katalogisiert werden.

Mit dem Compiler QPG konnen komplette Libraries, einzelne QPG-Programme oder Gruppen von Programmen umgewandelt werden:

```
// JOB COMPILE
// EXEC QPG,SIZE=AUTO
OPTIONS DISK PUNCH MVS UCTRAN LINES=66 CUU=xxx OBJ
COMPILE LIB=PROG PHASE=QPGPROG
REPLACE DOK=xxxxxxxx
REPLACE PROG=xxxxxxxx
REPLACE FROM=yyyyyyyy TO=zzzzzzzz
/*
// IF $RC NE 0 THEN
// GOTO ENDE
// ASSGN SYSIPT,cuu                    (cuu wie bei CPG-Umwandlung)
// EXEC LNKEDT
/* EXEC LNKEDT,PARM='AMODE=31,RMODE=ANY'        (in ESA Umgebungen)
/. ENDE
/&
```

Die Befehle OPTION(s), COMPILE und REPLACE müssen in dieser Reihenfolge angegeben werden und ab Stelle 1 beginnen. OPTION(s) und REPLACE sind wahlweise.

Wird die Programm-Library im CICS benutzt, so muss die Phase in der CICS-PPT eingetragen sein.

ESA-Mode

QPG kann in CICS- und in Batch-Umgebungen im ESA-Mode ausgeführt werden. Damit sind CICS- und Batch-Partitions unterstützt, die größer 16 MB sind. Bei der Installation wird QPG mit AMODE=31, RMODE=ANY gelinkt. Das gleiche gilt auch für eigene QPG-Libraries (s.o.). QPG wird dadurch bei der Ausführung in den erweiterten Adressraum oberhalb 16 MB geladen.

Libraries, die größer sind als 500 K-Byte, müssen mit AMODE=31 gelinkt werden. Es können QPG-Phasen erstellt werden, die maximal 2.000.000 Bytes groß sind.

OPTIONs definiert Standardwerte:

CUU=xxx	Angabe Reader-Adresse, Default ist READER.
DISK	Ausgabe auf Platte, Default (IJSYS04), muss jedoch eingetragen werden, wenn sowohl DISK als auch PUNCH benutzt werden.
DUMP	Abbruch (mit Speicherauszug) bei Fehlern.
LINES=nn	Anzahl Zeilen pro Seite in der Liste, Default=66.
MVS	bei Generierung für ein z/OS-System.
OBJ	wenn nur Object Code generiert werden soll.
PUNCH	Ausgabe auf Puncher, schaltet DISK aus.
UCTRAN	wenn die Liste übersetzt werden soll.
UPGRADE	automatisches New-Copy bei Ausführung geänderter Programme.

COMPILE definiert Umwandlungsparameter

LIB=xxxx	Angabe der QTF-Library, Default ist PROG
PHASE=yyyyyyyy	optional, Default ist QPGxxxx (xxxx=LIB)
SPACE=xxx	definiert den verfügbaren Platz für neue, zusätzliche Programme.

CREATE erstellt neue (leere) Libraries

Parameter wie bei COMPILE

REPLACE optional zur Umwandlung (Ersetzen) einzelner Programme

DOK=xxxxxxxx	gibt das QPG-Programmdokument an
FROM=yyyyyyyy	Gruppe der Dokumente ab, Default ist Anfang.
PROG=xxxxxxxx	wie DOK
TO=zzzzzzzz	Gruppe der Dokumente bis, Default bis Ende.

Mehrere Replace-Anweisungen können angegeben werden.

Die Umwandlungspartition muss mindestens 4,5 MB groß sein.

Hinweise

In der OPTIONS-Anweisung ist der Parameter UPGRADE unterstützt. Mit der Angabe von UPGRADE wird jetzt automatisch beim ersten Laden einer Programm-Library überprüft, welche Programme seit der letzten Umwandlung geändert wurden. Bei diesen Programmen wird dann automatisch ein New-Copy durchgeführt, damit bei der Ausführung die aktuelle Version benutzt wird. Mit dieser Option wird die Schnelligkeit des QPG-Compilers mit dem Vorteil der aktuellen Version des QPG-Generators kombiniert.

In der OPTIONS-Anweisung ist der Parameter DUMP unterstützt. Bei ersten Fehler in der Umwandlung wird der Compiler abgebrochen und ein DUMP erzeugt.

Es werden nur die Programme umgewandelt, bei denen im QTF als Dokumententype ' ' oder 'P' angegeben ist. Programme, die noch in der Entwicklung sind, lassen sich z.B. durch die Angabe 'T' (Test) von der Umwandlung ausschließen, so dass hier bei der Ausführung immer die neueste Version aktiviert wird.

Es ist die Anweisung CREATE unterstützt, um neue Libraries zu erstellen. CREATE wird anstelle von COMPILE benutzt, um eine leere Library zu formatieren.

Die Anweisungen COMPILE und CREATE können mit der Angabe SPACE=xxx benutzt werden. Dabei ist für xxx ein Wert von 0 bis 256 einzutragen. SPACE gibt den verfügbaren Platz für Programme an, die neu angelegt und zusätzlich in den Pool geladen werden. Fehlt die Angabe SPACE, dann werden bei COMPILE und bei CREATE 64K Bytes als Freiplatz für neue Programme reserviert. Wird für SPACE ein höherer Wert als 256 angegeben, so wird der Freiplatz automatisch auf 256K Bytes begrenzt. Eine mit COMPILE generierte Phase kann einschließlich SPACE maximal 500K Bytes groß werden.

Batch Utility

QPG-Programme können sowohl online als auch in einer Batch-Partition ausgeführt werden. Das Batch-Utiliy QPGUTIL erlaubt den Aufruf eines QPG-Programms in einer Batch-Partition. Durch QPGUTIL entfällt die Notwendigkeit, ein eigenes HL1-Batch-programm zu katalogisieren, um mit dem Befehl PROG (nur im festen Format) bzw. PROGRAM(M) im freien Format ein QPG-Programm aufzurufen.

Beispiel:

```
// JOB QPGUTIL
// EXEC QPGUTIL,SIZE=AUTO
PROGRAMM LIBR
PROGRAMM LIBR DEBG
/*
/&
```

Es ist eine Vorlaufkarte erforderlich, in der von Stelle 1-8 der Programmname und von Stelle 10-13 die Library eingetragen wird. Befindet sich das Programm in der Library PROG, dann kann der Eintrag der Library entfallen.

QPGUTIL unterstützt auch die Testhilfe im Batch. Ab Stelle 15 kann ein Code angegeben werden, unter dem die Stop-Conditions (mit Transaktion QPG) erfasst wurden. Der Trace wird im Batch auf einen Logfile (TS QPGL, maximal 10.000 Sätze) gespeichert und am Schluss der Verarbeitung gedruckt. Bei einem Abbruch der Verarbeitung wird der Logfile (ab CPG-3 Version 2.1) am Schluss im HL1-Batch-Dump ausgedruckt.

Startparameter

Programmen, die mit QPGUTIL aufgerufen werden, können in der JCL-Anweisung Startparameter mitgegeben werden. Im aufgerufenen Programm werden diese an das Feld PARMS übergeben, wenn es entsprechend definiert ist. Das Feld PARMS muss eine Länge von 256 Stellen haben und vor CPGEDS definiert sein, wenn CPGEDS benutzt wird.

Beispiel:

```
// JOB QPGUTIL
// EXEC QPGUTIL,SIZE=AUTO,PARM='JAHR=2000,REPORT=UMSATZ'
PROGRAMM LIBR
/*
/&
```

Das folgende Programm benutzt die Startparameter im Feld PARMS:

```
-d
         parms              256.      * startparameter
-c
         parms display
```

Übergabe von Daten

In der Vorlaufkarte von QPGUTIL können Daten ab Stelle 20 in einer Feldgruppe **PRDATA (maximal 61 * 1 Stelle)** übergeben werden.

```
* $$ JOB JNM=LCP,DISP=D,CLASS=A
* $$ LST CLASS=L,FNO=0112,FCB=FCB12,DEST=(*,PR)
* $$ PUN CLASS=L
// JOB PRINZ
// LOG
// EXEC QPGUTIL,SIZE=AUTO
PRDATA   TEST      PROP1
/* B     TEST      <=== PROGRAM DATA ===>
/*
/&
* $$ EOJ
```

Das folgende Programm benutzt die Übergabeparameter in der Feldgruppe PRDATA:

```
*-------------------------------------------------------------------
*       Aufruf QPGUTIL mit Daten                          TEST.PRDATA
*-------------------------------------------------------------------
-d
        prdata       10 *   1.                * program data
-i
        array prdata
                      1     8   ddname
                      9    10   ddtype
-c
        select prdata
        ddname dsply
```

Beachte:

- **PRDATA muss als Array von einstelligen Alphafeldern definiert sein, damit das SELECT im Input aus dem array prdata funktioniert.**

- **In der Input-Satzbestimmung muss mit dem Schlüsselwort ARRAY gearbeitet werden.**

Alternativ kann natürlich auch in der Datat Division mit einer Überlagerung mit ORG oder der Overlay-Technik gearbeitet werden, dann ist kein SELECT erforderlich!

Schnittstellen

QPG-Programme werden in vielen Bereichen der Lattwein-Software angewendet. Aufrufe mit und ohne Datenübergabe erfolgen aus folgenden Produkten:

CPG .... In jedem CPG-Programm oder HL1-Modul kann online und im Batch mit der Operation PROG (nur im festen Format) bzw. PROGRAM(M) im freien Format jedes QPG-Programm aufgerufen werden. Daten können zwischen CPG- und QPG-Programmen ausgetauscht werden.

QSF .... Bei MAP, MAPI und MAPD kann ein QPG-Programm zur Prüfung der Eingabefelder auf Plausibilität oder zum Einblenden von Help-Windows benutzt werden.

Query .. Jeder Query-Report kann sowohl online als auch im Batch auf die Daten zugreifen, die ein entsprechendes QPG-Programm aus bestehenden Files, Datasets, Storages oder Rechenoperationen zusammenträgt.

QTF .... Im Menü kann mit X (execute) jedes QPG-Programm direkt aufgerufen werden. Mit dem Programm QTFSORT lassen sich damit z.B. Dokumente sortieren.

In Textbausteinen erlauben QPG-Programme den einfachen Zugriff z.B. auf Adress- oder Personaldateien, um Anschriften oder Daten in Briefe oder Auswertungen einzufügen.

Bei Druckausgabe können vorhandene Datenbestände von QPG-Programmen analysiert und mit aktuellem Stand in die Dokumentation eingefügt werden.

In Hilfetexten kann ein QPG-Programm aufgerufen werden. Siehe HELP-Facility im QTF-Handbuch.

QLF .... Bei der LIST-Ausgabe können z.B. Nebenrechnungen mit QPG-Programmen in der Druckroutine ausgeführt und die Ergebnisse auf der Liste ausgegeben werden.

PROG-Aufruf in CPG-Programmen

Mit der Operation PROGRAM oder PROGRAM-VAR im CPG-Format oder PROG in der RPG-Syntax kann ein QPG-Programm als externes Unterprogramm aufgerufen werden:

```
        PROCEDURE DIVISION
                            PROGRAM HUGO
                            PROGRAM OTTO TEST
                            PROG-VAR VARPRG
```

```
     C* RPG SYNTAX
     C                      PROG HUGO
     C                      PROG OTTO      TEST
     C                      PROG           VARPRG
```

Es wird jeweils zuerst das QPG-Programm HUGO in der Library PROG aufgerufen, danach das Programm OTTO in der Library TEST und zuletzt ein variables Programm, dessen Name und gegebenenfalls Library im 32-Byte langen Feld VARPRG bereitgestellt werden müssen.

Das Feld VARPRG hat folgenden Aufbau:

```
     DATA DIVISION
            VARPRG      0 * 32.                 * Variable PROGRAM-Operation
            PROGN            8.                 * Variabler Programmname
            PRLIB            4.                 * Variabler Libraryname
            PREST           20.                 * Reserviert
```

```
     D* RPG SYNTAX
     D        VARPRG      0 32                  * Variable PROGRAM-Operation
     D        PROGN          8                  * Variabler Programmname
     D        PRLIB          4                  * Variabler Libraryname
     D        PREST         20                  * Reserviert
```

Die Operationen PROG/PROGRAM und PROG-VAR sind im CPG identisch mit den gleichen Operationen, die in QPG-Programmen ausgeführt werden können.

PROG-Aufruf in anderen Programmiersprachen

Im CICS kann ein LINK oder XCTL auf das Programm QPGPRG ausgeführt werden. Das rufende Programm muss eine TWA-Size von mindestens 3840 haben.

In der COMAREA wird angegeben, welche Operation benutzt wird (normalerweise PROG) und in welcher Library welches Programm ausgeführt wird. Die Stellen von 1 – 20 der COMAREA sind für QPG reserviert. Ab der Stelle 21 können Benutzerdaten übergeben werden. Diese werden im QPG-Programm mit SELECT CPGCOM gelesen und mit EDIT CPGCOM bereitgestellt.

LINK wird benutzt, wenn die Kontrolle an das aufrufende Programm zurückgegeben wird. Damit kann z.B. ein QPG-Programm aufgerufen werden, das prüft, welcher Benutzer angemeldet ist und welche Sachgebiete der Benutzer bearbeiten darf. Diese Informationen werden dann in der COMMON AREA zurückgegeben.

XCTL wird benutzt, wenn QPG die Kontrolle über die Transaktion erhält und eine Rückkehr in das rufende Programm nicht vorgesehen ist. Damit wird dann z.B. ein QPG-Anzeigeprogramm aufgerufen, mit dem durch einen Datenbestand geblättert wird.

Beispiel für ein Cobol-Programm, das mit LINK ein QPG-Programm aufruft:

```
// EXEC DFHECP1$,PARM='LANGLVL(2),APOST'
   CBL LIB,APOST,NOADV,NODYNAM,RENT,BUF(4096)
      *---------------------------------------------------------------
      *  DIESES PROGRAMM RUFT MIT LINK (EXPR) EIN QPG-PROGRAMM AUF
      *---------------------------------------------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. TSTC02.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  COMAREA.
           05  PRCMD  PIC X(4).
           05  PRLIB  PIC X(4).
           05  PROGN  PIC X(8).
           05  PREST  PIC X(184).
       PROCEDURE DIVISION.
           MOVE 'PROG' TO PRCMD.
           MOVE 'TEST' TO PRLIB.
           MOVE 'COBOL   ' TO PROGN.
           EXEC CICS LINK PROGRAM('QPGPRG')
                COMMAREA(COMAREA) LENGTH(200) END-EXEC.
           EXEC CICS SEND TEXT
                FROM(COMAREA) LENGTH(200) END-EXEC.
           EXEC CICS RETURN END-EXEC.
/*
```

Beispiel für ein Cobol-Programm, das mit XCTL ein QPG-Programm aufruft:

```
// EXEC DFHECP1$,PARM='LANGLVL(2),APOST'
   CBL LIB,APOST,NOADV,NODYNAM,RENT,BUF(4096)
      *---------------------------------------------------------------
      *  DIESES PROGRAMM RUFT MIT XCTL (EXITP) EIN QPG-PROGRAMM AUF
      *---------------------------------------------------------------
       IDENTIFICATION DIVISION.
       PROGRAM-ID. TSTCOBPR.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  COMAREA.
           05  PRCMD  PIC X(4).
           05  PRLIB  PIC X(4).
           05  PROGN  PIC X(8).
           05  PREST  PIC X(84).
       PROCEDURE DIVISION.
           MOVE 'PROG' TO PRCMD.
           MOVE 'TEST' TO PRLIB.
           MOVE 'CPGSTA  ' TO PROGN.
           EXEC CICS XCTL PROGRAM('QPGPRG')
                COMMAREA(COMAREA) LENGTH(100) END-EXEC.
           EXEC CICS RETURN END-EXEC.
/*
```

Aufbau der COMAREA:

Stelle	
1 - 4	Befehl (muss 'PROG' sein)
5 - 8	QPG-Library
9 - 16	QPG-Programm
17 - 20	Return Trans-Id
21 - ...	Benutzerdaten

Kompatibilität zu CPG und HL1

Bei QPG wurde größter Wert auf Kompatibilität zu CPG gelegt. Dennoch gibt es technische Regeln, die zu beachten sind, wenn QPG kompiliert, bzw. CPG-Programme interpretiert werden sollen.

Nur KSDS-Dateien, HL1- und QPG-Datasets, FIND-Tabellen und Temporary Storages können mit QPG direkt verarbeitet werden. Satzarten, die an sich logische Dateien in einem gemeinsamen VSAM-Cluster sind, werden mit READI verarbeitet.

QPG unterstützt nicht alle CPG-Befehle (siehe Operationen). Es wird nur die CPG-Syntax (also nicht der RPG-Formalismus) benutzt. Nur ein Statement je Zeile ist zugelassen.

Schalter sind bei QPG nicht unterstützt. Für Abfragen werden daher interne Felder verwendet, z.B. CPGFRC für das Ergebnis einer Dateioperation oder CPGMPF für die PF-Taste nach einer MAP-Operation. Die Programme werden daher automatisch strukturiert.

Ausgabe

UPDAT .. Bei QPG wird ein Satz über die Output Division geändert, falls diese angegeben ist. Ansonsten wird die Input Division benutzt.

WRITE .. Hinzufügen, Beschreibung analog zu UPDAT.

Schablonen in der Ausgabe sind im QPG nicht vorgesehen.

Konstanten

Konstanten können im QPG bis zu 72 Byte lang sein, im CPG jedoch nur 24 Byte. Das Ampersand '&' in Konstanten muss bei QPG nicht doppelt eingegeben werden. Im CPG müssen immer 2 Ampersands '&&' angegeben werden, da anschließend der Assembler hieraus 1 Ampersand macht.

Bsp:

QPG

```
-O
   FIELD SATZ
              2 '/&'
```

CPG

```
-O
    FIELD SATZ
               2 '/&&'
```

Variabler Storagename

CPGTSN muss im QPG-Programm definiert sein, damit die Verarbeitung kompatibel zu CPG erfolgt.

Operanden

Im CPG sind bei bestimmten Operationen nur Konstanten zugelassen, wie z.B. bei DELC, ELIM und FILL. Bei QPG sind hier sowohl Konstanten als auch Felder zugelassen.

Felder

Felder,die im Data Dictionary zentral definiert sind, können in QPG-Programmen ohne weitere Definition in der Data Division benutzt werden. Bei CPG muss hier mindestens der Feldname angegeben werden, wenn das Feld sonst im Programm nicht definiert wird.

Lange Feldnamen werden im CPG mit internen Namen (CPnnnn nnnn = Lfd-Nr.) durchnummeriert, wenn diese nicht im zentralen Data Dictionary definiert sind. Bei QPG wird der Name nach 6 Stellen abgeschnitten.

CPGSIN

Die Abfrage der Systeminformationen erfolgt bei QPG mit SELECT CPGSIN anstelle von COMRG CPGSIN.

Sprache

$-Zeichen für den Sprachencode in einem Map- oder List-Namen sind nicht im CPG unterstützt.

Data Division

Externe Felder sind nicht unterstützt.

Input und Output

Die Namen müssen bei QPG eindeutig sein.

Maximalwerte

	QPG	CPG	HL1
PWA-Größe	32.767	12.287	8.191
Satzlänge	32.767	9.999	9.999
Keylänge	256	99	99

Optimierung

Werden die LIST- oder MAP-Operationen verwendet, dann wird im QPG nicht optimiert. In CPG und HL1 werden nur die Felder definiert, die in den benutzten LIST-Dokumenten oder QSF-Maps gefunden wurden, wenn diese sonst im Programm nicht verwendet wurden.

Checkliste

Einige Regeln sind zu beachten, um mit QPG problemlos zu arbeiten:

EXHM ... Aufruf nur von Modulen mit OPTIONS DATaset oder PWA, wenn diese Dateizugriffe ausführen. EXHM aktualisiert normalerweise nicht das Feld CPGMPF, wenn im Modul eine Map (im Dialog) gelesen wurde. Mit OPTIONS PFK wird CPGMPF auch nach jeder EXHM-Operation aktualisiert.

FILES .. Dateien müssen im Data Dictionary definiert sein. Dateizugriffe erfordern OPTIONS DATaset oder PWA.

PROG ... Aufruf nur aus ESA-fähigen Commandlevel Programmen. Die maximale Größe des generierten Codes ist 32000 Bytes, das sind ca. 2.000 Operationen. Dennoch sollten Programme in kleine modulare Einheiten zerteilt werden.

SYNTAX . Nur ein Statement je Zeile. Code von Spalte 1-70 bzw. von Spalte 7-79 bei '-' in Spalte 6. Statements werden mit '.' beendet, dahinter kann ein Kommentar (mit '*') stehen.

EINGABE Satzarten müssen mit SEGMENT definiert sein und werden mit der Operation READI verarbeitet.

In bestimmten Fällen kann es erwünscht sein, ein mit QPG entwickeltes Programm nach der Testphase mit CPG oder HL1 umzuwandeln, z.B. um bei SQL-Verarbeitung dynamische durch statische Befehle zu ersetzen. Für die Umwandlung ist dann die OPTIONS-Anweisung um die für CPG oder HL1 benötigten Parameter zu ergänzen.

Die Operation TASK muss durch EXITT erstetzt werden, wobei der Ablauf dann im CPG-Programm zu steuern ist.

TWA-LOAD und TWA-SAVE sind nur in CPG-Hautprogrammen unterstützt.

DELC mit Service A ist im CPG in einer DO-Schleife zu codieren.

SETLL für Temporary Storage ist im CPG nicht vorgesehen, statt dessen kann bei UPDATE in Faktor 1 eine Satz-Nr. mitgegeben werden.

LIST und MAPx mit einem '$' Zeichen im List- oder Mapnamen sind im CPG nicht vorgesehen; der Sprachencode muss hier z.B. vom Programm in einem Feld für die variable LIST- oder MAP-Operation bereitgestellt werden.

Im QPG können bis zu 6 Stellen lange Index-Feldnamen benutzt werden; im CPG sind entsprechend kürzere Namen zu verwenden.

Aufruf im QSF

Mit dem DEFine Command kann im QSF ein QPG-Programm definiert werden, das automatisch aufgerufen wird, wenn zu dieser Map eine MAP-, MAPD- oder MAPI-Operation ausgeführt wird.

Im folgenden Beispiel wird das QPG-Programm QPGWS5T in der Library TEST aufgerufen:

```
   QQQQQ                                    V.L  UID  TERM  tt.mm.jj  ss.mmUHR
 QQ     QQ       Q üry
QQ       QQ       U ser
QQ       QQ         I nformation
QQ    QQ QQ         C ontrol
 QQ    QQQ            K it
   QQQQQ QQ                                              Quick Screen Facility
--------------------------------------------------------------------------------

QSF System Variable für Bild: PRQPGWS5

In diesem Bild können die Standardwerte für das oben genannte Bild
gesetzt werden. Ab der Eingabe werden diese Werte bei der Pflege des
Bildes anstatt Blank gesetzt.

Hilfe bei Taste                      Maske
Konstante Attribut   _  Farbe   _    E-H Wert
Variable  Attribut   _  Farbe   _    E-H Wert
Ausführung Auxilary
Prüfen QPG-Programm QPGWS5T   TEST
Hintergrund Bilder   ________  ________  ________

--------------------------------------------------------------------------------
PF1 ==> Hilfe                                                  PF12 ==> Ende
```

Das folgende Programm QPGWS5T wird z.B. benutzt, um zu prüfen, ob eine gültige Funktionstaste betätigt wurde:

```
-D
         INFO              79
-C
         IF CPGMPF = 'CL' OR
         IF CPGMPF = 'DE' OR
         IF CPGMPF = 'P2' OR
         IF CPGMPF = 'P3' OR
         IF CPGMPF = 'P7' OR
         IF CPGMPF = 'P8'
            FILL ' ' TO INFO
         ELSE
            INFO = 'Falsche Taste'
         END
```

Query-Programme

Mit der Enwicklung der Query-Programme 1982-1984 wurde ein Werkzeug geschaffen, das aus den damals bestehenden VSAM-Datenbeständen in einfachster Weise die benötigten Informationen selektieren, verdichten und zu einem Report aufbereiten konnte.

Die zahlreichen Wünsche wie sortieren (auch online), individuelle Listformate (mit QTF-LIST) und andere wurden weitgehend realisiert. Rechenoperationen konnten jedoch aus technischen Gründen nicht direkt in Query eingebaut werden.

Mit der Entwicklung von QPG wurde Query um ein Interface zum QPG erweitert. QPG erlaubt es, die benötigten Rechenoperationen, Dateizugriffe und Abfragen für Query interaktiv zu definieren.

Mit QPG wird ein Programm für Query erstellt. Die logische Datensicht (View) des Query-Programms muss dabei, wie bei Query üblich, im Data Dictionary definiert sein. Zweckmäßig werden dabei die DD-Einträge der gewünschten Dateien und Satzarten in die neue Struktur kopiert. Dabei sollten nur die tatsächlich benötigten Felder ausgewählt werden. In der View des Programms werden dann noch die erforderlichen Arbeits- und Rechenfelder definiert. Wie beim Datenkanal von HL1-Modulen können in der View numerische Felder nur gepackt verarbeitet werden.

Die Programm-View erhält im Data Dictionary ein eigenes Organisationskennzeichen (P) und wird mit Einheit PROG beschrieben. Zum Speichern des zum Query gehörenden Programms wird die QTF-Library PROG verwendet. Soll z. B. aus organisatorischen Gründen eine andere Library benutzt werden, so ist diese Library bei Einheit im DD anzugeben.

Beim Erstellen eines Query-Reports wird als Dateiname der View-Name des Programms angegeben. Query erkennt dann automatisch, dass hierzu ein Programm mit QTF generiert werden muss. Durch Bedienerführung wird am Bildschirm voll automatisch ein Rahmenprogramm für Query in der QTF-Programm-Library erstellt. Dieses Rahmenprogramm ist funktionsfähig und kann direkt ausgeführt werden, nachdem mit TPQB der Query-Report beschrieben wurde. Das Programm kann jederzeit geändert und erweitert werden. Es ist so z.B. möglich, bei der CICS-Ausführung eine Map einzublenden, durch die vom Benutzer während der Ausführung Selektionskriterien am Bildschirm eingegeben werden können.

Das QPG-Programm wird mit TPQB und der entsprechenden Programmfunktion gepflegt. Bei einer Änderung wird automatisch ein Newcopy durchgeführt. Durch das QPG-Programm wird Query so flexibel, dass fast alle Wünsche realisiert werden können.

Checkliste für Query-Programme

DD

Datei mit dem gewünschten Programmnamen anlegen. Dateiorganisation 'P' und Einheit 'PROG'. Wird statt PROG eine andere QTF-Library verwendet, so ist diese bei Einheit anzugeben. In das DD werden alle gewünschten Felder aus den zu verarbeitenden Dateien und egebenenfalls Satzarten kopiert, die vom Query verarbeitet werden sollen oder dürfen. Dahinter werden eventuell benötigte Arbeitsfelder für Zwischen- oder Endergebnisse angelegt.

In der Dateibeschreibung ist eine Keylänge einzutragen, wenn im Query die Funktionen KEY oder EKEY benutzt werden.

TPQB

Programm anlegen, mit Dateiname = Dokumentname. Eingabe Datei(en) und gegebenenfalls Satzart(en). Rechenformeln.

PROG

Query überträgt bei Programmaufruf im Feld CPGFRC einen Operationscode. Dieser Code ist abhängig von der Umgebung, in der Query ausgeführt wird.

	BATCH		CICS
'O'	Open	'S'	Setll
'R'	Read	'R'	Read
'C'	Close	'T'	Tclose
		'D'	Random

Open, Close, Setll und Random (O,C,S,D) werden jeweils nur einmal zu Beginn oder zum Ende des Query Reports aktiviert. Read (R) fordert so lange Daten an, wie diese von Query benötigt werden. Tclose (T) bietet vor jeder Query-Terminal-Anzeige die Möglichkeit, Bereiche (wie z.B. VSAM-Strings) freizugeben. Beim nächsten Read muss dann neu positioniert werden.

CPGFRC übergibt den Return-Code an Query:

'EF' End of File. Hierdurch wird Query beendet.

Das folgende QPG-Programm wird von Query aufgerufen, um die Datei CPGKDN zu verarbeiten und um Rechenoperationen durchzuführen:

```
*-------------------------------------------------------------------
*        Query dataset SEMKDN                              TEST.SEMKDN
*-------------------------------------------------------------------
options  dataset.
         file CPGKDN inp.            * physical file
data division.
  define SEMKDN.                     * fields for Query
         opcode              1.      * operation code
         cpgkey             64.      * Query key
input division.
         file CPGKDN dd.
procedure division.
         opcode = cpgfrc.
         evaluate.
         when opcode = 'O'.          * batch open
              open CPGKDN.
              cpgkey setll CPGKDN.
         when opcode = 'S'.          * online setll
              cpgkey setll CPGKDN.
         when opcode = 'R'.          * read batch and cics
              read CPGKDN.
              RF2 = 1.
         when opcode = 'T'.          * online tclose
              read  CPGKDN.
              rndom CPGKDN.
         when opcode = 'D'.          * online rndom
              rndom CPGKDN.
         when opcode = 'C'.          * batch close
              close CPGKDN.
         end-evaluate.
```

QPG-Programme in QTF-Textbausteinen

Mit dem Befehl §PROGRAM kann beim Einfügen eines QTF-Textbausteines ein QPG-Programm aufgerufen werden. Das QPG-Programm kann dann z.B. die gewünschte Adresse aus der Kunden- oder Lieferantendatei lesen und als variable Daten dem Textsystem wieder zur Verfügung stellen.

Das folgende Beispiel zeigt den Textbaustein BRIEF, der mit dem Befehl §PROGRAM das QPG-Programm BRIEF in der Library PROG aufruft. Dieses Programm blendet eine QSF-Map ein, in der eine Kundennummer eingegeben wird. Hiermit wird dann die Anschrift mit CHAIN aus der Datei geholt. Die Anschrift wird mit dem Standard-List-Dokument BRIEF auf Temporary Storage zwischengespeichert.

Der TS-Arbeitsspeicher wird mit dem Befehl §LOADQUE im Textbaustein BRIEF in das Dokument eingefügt.

Textbaustein BRIEF

```
§PROGRAM BRIEF
§LOADQUE QTFP
```

Felder aus dem Editor

Folgende Felder können im Programm definiert werden, um beim §PROGRAM-Aufruf Informationen vom Editor zu übernehmen, bzw. an den Editor zurückzugeben:

```
-D
        DOKUM              8.      * Dokumentname
        DKLIB              4.      * Library
        IOLINE           256.      * Textzeile mit §PROGRAM Befehl
        INFO              79.      * Evtl. Fehlermeldung für QTF-Editor
```

QPG-Programm BRIEF

```
OPTIONS  DAT DEF
         FILE CPGTST                    * Adressdatei
-D
         DEFINE SQLADR                  * Variable Daten
-I
         FILE CPGTST DD REF SQLADR.     * Felder aus Data Dictionary
-C
         MAPD BRIEFA.                   * Kundennr. eingeben
         KEY = '00000'
         MOVE KDNRA KEY.
         KEY CHAIN CPGTST.              * Direktzugriff auf die Daten
         IF CPGFRC = 'NF'
            EM000 = 'NICHT GEFUNDEN'.   * Fehlermeldung, wenn nicht gefunden
         END
         MAPD BRIEFE LOW.               * Daten anzeigen und ggfs. ändern
         IF CPGMPF = 'DE'
            'QTFP' LIST BRIEF HEADER.   * Brief auf TS zwischenspeichern
         END
```

QTF LIST-Dokument BRIEF

```
§section header



§Firma
§section header    §if hoderf = 'H'
Herrn §name
§section header    §if hoderf = 'F'
Frau §name
§section header    §if hoderf = ' '
§name
§section header
§str

§plz §ort


                                        Düren, den §udate


§btreff



§section header    §if hoderf = 'H'
Sehr geehrter Herr §name_,
§section header    §if hoderf = 'F'
Sehr geehrte Frau §name_,
§section header    §if hoderf = ' '
Sehr geehrte Damen und Herren,
§section header

...    T e x t     ...


Mit freundlichen Grüßen

Lattwein GmbH

i.A.


§vname_ §nname
```

Programmaufruf beim Drucken mit QTF

Mit dem Befehl §PROGRAM wird beim Drucken eines Dokumentes mit QTF ein QPG-Programm aufgerufen. Hiermit können aktuelle Daten an der entsprechenden Stelle in die Liste eingefügt werden.

Es soll z.B. die QTF-Druckertabelle mit folgendem Dokument gedruckt werden:

```
Im QTF sind folgende Drucker angelegt:

§PROGRAM PRINTER        <--- Dieses Programm listet die QTF Druckertabelle auf

:
```

Das QPG-Programm PRINTER liest die QTF-Druckertabelle (Satzart 02) mit dem Dataset HQTFF aus der Textdatei QTFTXT. Dabei werden die Felder CPGDID, DOKUM, LIBR und IOLINE von QTF zur Verfügung gestellt:

```
OPTIONS  DEFINE DATASET
         FILE HQTFF
-D
               CPGDID        4.        * Drucker-Id aus QTF
               DOKUM         8.        * Document aus QTF
               LIBR          4.        * Library aus QTF
               IOLINE      256.        * Command Line aus QTF
               VS            1.        * Vorschub
-I
         FILE HQTFF HS
                         1    4   CPGHIC
                        11   52   SATZ
         FIELD SATZ
                         3    6   DRID.           * Druckername
                        21   28   VORS.           * Vorschub
                        29   29   EXIT.           * Drucker-Exit
                        30   30   RESV.           * Reservierungstest
               BIN      31   32 0 SHIFT.          * Verschiebung
                        33   36   ALTDR.          * Alternativer Drucker
                        37   42   TYPE.           * Printer Type
-C
         CPGDID LIST PRINTER HEADER
         SATZ = '02  '
         SETLL HQTFF
         DO LOOP
            READ HQTFF
            IF SATZ >< '02'
               BREAK
            END
            SELCT SATZ
            VS = '0'
            IF VORS = X'0006'
               VS = 1
            END
            IF VORS = X'0007'
               VS = 2
            END
            IF VORS = X'0008'
               VS = 3
            END
            CPGDID LIST PRINTER DETAIL
         END
         RNDOM HQTFF
         CPGDID LIST PRINTER TRAILR
```

Das LIST-Dokument PRINTER beschreibt das Ausgabeformat für die QTF Druckertabelle:

```
§section header
Verzeichnis der QTF Printer

drid     exit      res-test        shift         alt.dr    type      v
----     ----      --------        -----         ------    ------    -
§section detail
§nametab drid,exit,resv,shift:zr,altdr,type,vs
%%%%     %         %                 %%%         %%%%      %%%%%%    %
§section trailr
-------------------------------------------------------------------------

Ende der Printer Tabelle
```

Es ergibt sich z.B. folgender Ausdruck:

Im QTF sind folgende Drucker angelegt:

Verzeichnis der QTF Printer

drid	exit	res-test	shift	alt.dr	type	v
CAEF	T		7	LW07		1
CAET	T	X		LW07		1
GB		X	10	L86C		1
GB2		X	11	L86O		1
IPDD	I			L86C	IPDS	1
IPDS	S				IPDS	1
JOB	X			PRDR		0
L86C				L86C		1
L86O			9			1
PDOK	X			DUMY		1
PLST	X	X				1
PLS2	X	X				1
PPUN	X					1
PRDR	X					1
PRDS	X					1
PRIN						1
PRT1	X					1
PSTR	X					0
QTFD	S					1
QTFI	S					1
QTFP	S		1			1
QTFS	S					1
QTFY	S					1
SUBH	X					1
SUBP	X			SUBH		1

Ende der Printer Tabelle

Programmaufruf bei LIST-Verarbeitung

Der Befehl §PROGRAM ruft bei der LIST Verarbeitung ein QPG-Programm auf. Hiermit können z.B. Zwischenrechnungen durchgeführt werden, und die Ergebnisse in die Liste eingefügt werden.

Im folgenden Beispiel wird eine Jahresabrechnung erstellt. Das LIST-Dokument ABRG enthält das gewünschte Ausgabeformat:

```
§section detail

:

Für das Jahr §jahr ist folgender Betrag fällig:            §betr:j        €

§section detail    §if betr < 100

Bitte zahlen Sie den fälligen Betrag bis spätestens 30.Juni.§jahr

§section detail    §if betr >= 100
§PROGRAM ABRG                          <--- Unterprogramm Abrechnung
Bitte zahlen Sie folgende Teilbeträge:

bis zum 30.März.§jahr                                      §btr1:j        €

bis zum 30.Juni.§jahr                                      §btr1:j        €

bis zum 30.Sept.§jahr                                      §btr1:j        €

bis zum 30.Dez. §jahr                                      §btr2:j        €

§PRETURN                               <--- Zurück zum Hauptprogramm

:
```

Ist der verbleibende Betrag >= 100 €, so kann er in 4 Teilen bezahlt werden. In diesem Fall wird das QPG-Unterprogramm ABRG in der Library PROG aufgerufen.

Hinweis: Befindet sich das Programm nicht in der Library PROG, dann ist die Library in Klammern hinter dem Dokumentnamen anzugeben, z.B.:

```
§PROGRAM ABRG(TEST)
```

QPG-Programm ABRG:

```
         OPTIONS DAT
-D
         BETR     11 2
         BTR1     11 2
         BTR2     11 2
         JAHR      4
-C
         BTR1      =   BETR / 4
         MOVE '000' TO BTR1.               * Abrunden auf volle 10 €
         COMPUTE BTR2 = BETR - 3 * BTR1.   * Restbetrag letztes Quartal
```

Das Programm ermittelt die Teilbeträge BTR1 und BTR2, dabei wird BTR1 auf volle 10 € gerundet und BTR2 enthält den verbleibenden Restbetrag. Das Feld JAHR wird definiert, weil es in diesem Abschnitt der Liste ausgegeben wird.

QPG-Datasets

QPG-Datasets werden wie HL1-Datasets mit den Dateioperationen READ, RNDOM, CHAIN usw. aufgerufen. Beim Aufruf wird der Operationscode im Feld CPGFRC übergeben. Der Operationscode entspricht den Stellen von 1-2 im Feld CPGHIC bei HL1-Datasets. Der Returncode wird in CPGFRC zurückgegeben.

HL1-Module und HL1-Datasets

Alle Funktionen, die mit QPG nicht direkt ausgeführt werden können, lassen sich mit Hilfe von HL1-Modulen realisieren, die mit dem Befehl EXHM oder als Datasets mit den Dateioperationen aufgerufen werden.

Achtung:
Die HL1-Tabellen müssen in der CICS-PPT definiert sein. HL1- Module mit Dateizugriffen müssen OPTIONS DATaset oder PWA enthalten.

Mit HL1-Datasets ist der Zugriff zu jeder beliebigen Datenbank möglich. Das Feld CPGHIC muss im Datenkanal vorhanden sein. Der Returncode wird im Feld CPGFRC automatisch bereitgestellt und ist nach jedem Zugriff zu analysieren. CPGFRC darf nicht im Datenkanal enthalten sein.

Beispiel:

```
-I
     FILE KANAL HS
                    1   8   DOKUM
     FILE HQFFM HS.
                    1   4   CPGHIC.
                    5  12   MAP.
-C
     EXHM HHELP KANAL.             * Aufruf Help Dokument
     DO LOOP
         READ HQFFM.
         IF CPCFRC = 'EF'.         * End of File
             RNDOM HQFFM.
             BREAK
         END
         :                         * Verarbeitung
     END
```

Eigene Trans-Ids für QPG-Programme

Mit QPGSTRT kann jedem QPG-Programm eine eigene Trans-Id gegeben werden. Hiermit kann es im CICS per Tastatur oder aus einer Anwendung, die z.B. in einer anderen Programmiersprache erstellt wurde, per Start Trans-Id einfach aufgerufen werden.

In der PCT ist für die gewünschte Trans-Id das Programm QPGSTRT einzutragen. Bei Start der Trans-Id wird das QPG-Programm START in der Library TASK aufgerufen. Dieses Programm kann in jedem Unternehmen nach eigenen Bedürfnissen erstellt werden und dient als Verteiler für die Anwendungsprogramme. Ein Muster wird unter QPGDSTRT in der PROG-Library ausgeliefert. Das Musterprogramm verzweigt beispielsweise in ein QPG-Anwendungsprogramm mit dem Namen der Trans-Id in der Library TASK.

Beispiel:

```
*---------------------------------------------------------------------
*        start transaction with qpgstrt                    TASK.START
*---------------------------------------------------------------------
-d
        trid                 4.      * transaction id
        vprog               32.      * variable program call
        xcomrg              32.      * communication region
-i
        field xcomrg
                         7   10   trid
-c
        comrg xcomrg.                * get communication region
        selct xcomrg.                * find trans-id
        vprog = trid.                * use it as program name
        prog-var vprog.              * call program
```

In QPG verwendete Storages

Name	Q	Recl	Beschreibung (Q=Queue)
QPGA	Q	32	Aufruf aus Userprogramm
QPGC	Q	32	internal Calls
QPGD	Q	var	Save Terminal bei Debug
QPGF	Q	8.152	Compressed File Storage
QPGL	Q	100	Logfile bei Debug
QPGT	Q	104	Trace Conditions bei Debug
QPGV	Q	64	Dokumentenverzeichnis
QPGW	Q	2.048	Working Storage

Beispiele

Dieses Kapitel enthält einige Musterprogramme, die Anwendungsmöglichkeiten von QPG-Programmen zeigen.

Tischrechner

Das folgende QPG-Programm erlaubt es, einen Tischrechner zu simulieren und aus jeder CPG-Anwendung mit dem Befehl PROG aufzurufen:

```
*--------------------------------------------------------------------
*        Tischrechner                                      PROG.PRTIRE
*--------------------------------------------------------------------
-d
         erg       11 2
         f2        11 2
         op         1
-c
         twa-load
         map  prtire
         evaluate
         when cpgmpf = 'P3'
              exiti 'QTF'
         when op = '+'
              erg = erg + f2
         when op = '-'
              erg = erg - f2
         when op = '*'
              erg = erg * f2
         when op = '/' and
              f2 >< 0
              erg = erg / f2
         end-evaluate
         mapo prtire
         twa-save
         task
```

Die Map PRTIRE enthält die Felder F2 und OP (Eingabe) und ERG (nur Ausgabe).

Compile-Prog

Das folgende QPG-Programm erlaubt es, im VSE einen Job in die Power-Reader Queue zu submitten. Der Compile-Job dient dazu, eine QPG-Programmlibrary (Feld PRLIB) als Phase zu katalogisieren.

```
-D
         PRLIB               4
         INFO               79
-C
         IF PRLIB = ' '
            PRLIB = 'QXFS'
         END
         'PRDR' LIST COMPILE
         INFO = 'Compile Job submitted'
```

Compile-Job

Der Job ist als QTF-LIST-Dokument in der Library JOB gespeichert:

```
* $$ JOB JNM=LCL,CLASS=A
* $$ PRT CLASS=L,FNO=0112,FCB=FCB12
// JOB COMPILE
// EXEC QPG,SIZE=AUTO
OPTIONS DISK
COMPILE LIB=§PRLIB
/*
// IF $RC NE 0 THEN
// GOTO ENDE
// ASSGN SYSIN,cuu
// EXEC LNKEDT
/. ENDE
/&
* $$ EOJ
```

Inhaltsverzeichnis QTF-Dokument

Das folgende QPG-Programm zeigt aus einem QTF-Dokument jeweils die erste Zeile (Überschrift) jeder Seite an. Bei entsprechend organisierten Dokumenten kann damit ein Inhaltsverzeichnis am Bildschirm dargestellt werden.

```
options  dataset
         file hqtfc
-d
         x                    3 0
         page        20 *   80
-i
         file hqtfc hs dd
-c
         dokum = 'QXFHEL  '
         libr  = 'QXF '
         open hqtfc inp
         do 20 with x
            seite = x
            zeile = 1
            chain hqtfc
            if cpgfrc = 'NF'
               break
            end
            page(x) = satz
         end
         mapd page
         close hqtfc
```

QPG-Dataset

```
options  dataset.                    * Wissensbanken in CPGWKV
         file CPGWKV type xk.        * define file
-d.
         record        0 *  98.      * data record
         cpgfrc             2.      * file return code
  define qxfklb.                     * define fields
         cpgeds        0 *   4.      * ---------------------------------
         opcode             2.      * operation code
         savrec            98.      * save record
-i.
         file CPGWKV dd type xk.     * physical file
                       3    10   key
-c.
          opcode = cpgfrc.           * move operation code
          cpgfrc = '  '.             * init return code
          savrec = record.           * save record
          edit keyv.                 * edit key vsam
          evaluate
             when opcode = 'Z'.      * check
                check CPGWKV
             when opcode = 'O'.      * open
                open CPGWKV
             when opcode = 'G '.     * chain
                keyv chain CPGWKV
             when opcode = 'GC'.     * chain - c
                keyv chain CPGWKV check
             when opcode = 'GP'.     * chain - p
                keyv chain CPGWKV prot
             when opcode = 'GU'.     * chain - u
                keyv chain CPGWKV upd
             when opcode = 'S'.      * setll
                keyv setll CPGWKV
             when opcode = 'R'.      * read
                keyv read CPGWKV
                if keyv >< '42'
                   cpgfrc = 'EF'
                   record = savrec
                end
             when opcode = 'B'.      * readb
                keyv readb CPGWKV
                if keyv >< '42'
                   cpgfrc = 'EF'
                   record = savrec
                end
             when opcode = 'D'.      * rndom
                rndom CPGWKV
             when opcode = 'U'.      * updat
                updat CPGWKV
             when opcode = 'N'.      * write
                write CPGWKV
             when opcode = 'L'.      * delet
                keyv delet CPGWKV
             when opcode = 'C'.      * close
                close CPGWKV
          end-evaluate
-o
          field keyv
                              2 '42'
                key           10
```

Im DD ist die physische Datei CPGWKV, Satzart XK beschrieben:

D a t a D i c t i o n a r y Datei: CPGWKV Sa: XK 26.02.96 11.30UHR

Kz: Prog-Kz: LW

---> FCPGWKV U 4024 20KV DISK

Feldname	von	bis	P	Länge	FGE	Key	E	Beschreibung
KEYV	1	20		20		Y		SCHLUESSEL DER CPGWKV
AO	21	21		1				AND OR
C123	22	24		3				BEDINGUNG 1 2 3
OP1	25	32		8				OPERAND 1
DY1	33	34		2				DUMMY1
OP	35	36		2				OPERATOR
DY2	37	39		3				DUMMY 2
OP2	40	47		8				OPERAND 2
DY3	48	49		2				DUMMY 3
OP3	50	57		8				OPERAND 3
SB	58	58		1				SERVICE BYTE
S123	59	61		3				SETZEN BEDINGUNG 1 2 3
TS	62	62		1				STATEMENT TYP
TF1	63	63		1				TYP FAKTOR 1
TF2	64	64		1				TYP FAKTOR 2
TG	65	65		1				ZIEL ART
TEXT	66	121		56				BESCHREIBUNG
TEXTA	122	130		9				TEXT RESERVE

Ende des Datenbestandes.

Im DD ist die logische Datei (Dataset) AKBD ebenfalls beschrieben:

D a t a D i c t i o n a r y Datei: AKBD Sa: 26.02.96 11.32UHR

Kz: Prog-Kz: PR qag wissensbanken dataset

---> FAKBD U 256 8 P TEST

Feldname	von	bis	P	Länge	FGE	Key	E	Beschreibung
KEY	1	8		8		Y		schlüssel
KNA	1	2		2				WISSENSBANK NUMMER
FNA	3	4		2				FOCUS NUMMER
RNA	5	6		2				REGEL NUMMER
SNA	7	8		2				STATEMENT NUMMER
OP1	9	16		8				OPERAND 1
OP	17	18		2				OPERATOR
OP2	19	26		8				OPERAND 2
TS	27	27		1				STATEMENT TYP
TF1	28	28		1				TYP FAKTOR 1
TF2	29	29		1				TYP FAKTOR 2
TG	30	30		1				ZIEL ART
OP3	31	38		8				OPERAND 3
GNA	39	40		2				ZIEL NUMMER ALPHA
TEXT	41	96		56				BESCHREIBUNG

Ende des Datenbestandes.

Suchen in Tabellen

```
options  dat
         FILE CPGKDN.                  * KUNDEN
         FILE ksuc.                    * FIND-TABELLE
-D.
         DEFINE ksuc.
         case 1 0
         cpgeds 0 * 1.                 * end of data segment
-I.
         FILE CPGKDN DD.               * KUNDEN
         FILE ksuc DD.                 * TABELLE
-C.
         twa-load.
         if cpgfrc = 'EF'
            mapo prsk
            uctran off task
            twa-save.
            task.
         end
         map prsk low.
         rndom ksuc
         if kdnra > ' '
            kdnra find ksuc
         end
         if case = 0
            evaluate
               when firma2 > ' '
                  case = 1
               when kname > ' '
                  case = 2
               when ort2 > ' '
                  case = 3
               when plz2 > ' '
                  case = 4
            end-evaluate
         end
         evaluate
            when case = 1
               firma2 find ksuc
            when case = 2
               kname find ksuc
            when case = 3
               ort2 find ksuc
            when case = 4
               plz2 find ksuc

         end-evaluate
         if cpgfrc = '  '.
            kdnra chain cpgkdn
            mapo prkdmp3a
            twa-save
            task
         else
            fill ' ' to kdnra
            case = 0
            mapo prsk
            twa-save
            task
         end
```

In der Maske PRSK werden die Suchbegriffe eingegeben:

```
prsk

firma2  %firma2

kname   %kname

ort2    %ort2

pl2     %plz2
```

Je nach Eingabe wird die Tabelle KSUC mit der Operation FIND durchsucht nach den ersten 2 Stellen des Firmannamens, dem Kurznamen, den ersten 2 Stellen des Orts oder den ersten 2 Stellen der Postleitzahl.

Wird in der Tabelle ein Eintrag gefunden, dann wird mit der Kundennummer (KDNRA) die Kundendatei CPGKDN gekettet und die Daten des Kunden werden angezeigt. Im Anzeigemode wird mit Betätigen der (Enter-) Taste weitergeblättert, indem mit der letzten Kundennummer positioniert wird (KDNRA FIND KSUC) und danach mit dem Suchbegriff wieder eine FIND-Operation ausgeführt wird.

Der Ablauf ist taskorientiert. Nach einer MAP-Ausgabe wird das Programm mit der Operation TASK erneut (taskorientiert) gestartet. Die Operationen TWA-SAVE und TWA-LOAD dienen dazu, die Felder bis CPGEDS in TS zwischenzuspeichern.

Mit EVALUATE wird gesteuert, welche FIND-Operation jeweils durchgeführt wird, d.h. welcher Suchbegriff benutzt wird. Da die Eingabefelder mit FIND aus der Tabelle gelesen werden, können diese beim Blättern nicht benutzt werden. Aus diesem Grund wird für die Steuerung ein Hilfsfeld (CASE) benutzt.

RRDS- und ESDS-Dateien

Für diese VSAM-Dateien gelten bestimmte Regeln, Beispiel:

```
options  dat def
         file cpgesd.                * esds file
         file cpgrrd.                * rrds file
-d
         rrn               9 0.      * rel. record number
-i
         file cpgesd
                       1   80   satz
         field cpgk01
               bin     1    4 0 rba
         file cpgrrd
                       1   80   satz
-c
         debug on
         rba = 0
         do 10
            rba read cpgesd
            selct cpgk01
         end
         rrn = 1
         rrn read cpgrrd
         do 10
            rrn read cpgrrd
         end
```

Bei ESDS-Dateien muss eine gültige RBA bei der ersten READ- oder CHAIN-Operation angegeben werden. Nach einer READ- oder WRITE-Operation kann die aktuelle RBA mit einer SELCT-Operation aus dem internen Key-Feld CPGKxx (xx = Nr. der FILE-Anweisung, also 01 = 1. Datei) geholt werden.

Auf RRDS-Dateien muss bei READ oder CHAIN mit einer gültigen relativen Satznummer zugegriffen werden.

Als Key für RRDS- und ESDS-Dateien kann auch jeweils ein 4 Bytes langes Alphafeld dienen, das entsprechend binär aufbereitet ist.

Ansonsten gelten bei ESDS- und RRDS- Dateien die gleichen Regeln wie bei VSAM-KSDS-Dateien.

Variable Satzlänge

Bei VSAM-Dateien und Temporary Storages mit variabler Satzlänge wird mit dem Feld CPGVRL nach einer READ- oder CHAIN-Operation die tatsächliche Satzlänge zur Verfügung gestellt. Bei WRITE- und UPDAT-Operationen kann CPGVRL außerdem benutzt werden, um die Ausgabesatzlänge vorzugeben. Hierzu wird CPGVRL vor WRITE und UPDAT auf den gewünschten Wert gesetzt, und die Datei wird in der Output Division mit dem Parameter VAR beschrieben:

```
options  dat
         file cpgksd
-d
         cpgvrl            5 0.      * var. record length
-i
         file cpgksd
                      1   24   key
                     25  280   daten
-c
         key = 'TEST-CPGVRL'
         fill '*' to daten
         key chain cpgksd p
         cpgvrl = 100
         if cpgfrc = 'NF'
            write cpgksd
         else
            updat cpgksd
         end
         exiti 'TPDI'
-o
         file cpgksd var
                     key    24
                     daten 280
```

Batch-Returncode

In einem QPG-Programm kann in dem internen Feld CPGEOJ ein Returncode für die Job-Control-Steuerung übergeben werden.

Beispiele:

```
procedure division.                 * Setzen JCL Variable $RC
          .
          CPGEOJ = 0.               * Alles ok
          .
          CPGEOJ = 4.               * Warning
          .
          CPGEOJ = 8.               * Fehler
          .
```

Der zuletzt gesetzte Returncode ist gültig.

Im Batchjob kann dann der Returncode $RC abgefragt werden. Beispiel:

```
// JOB CPGEOJ
// EXEC QPGUTIL,SIZE=AUTO
TPREOJ   TEST
/*
// IF $RC = 0 THEN
* ALLES OK: DER RETURNCODE IST 0
// IF $RC = 4 THEN
* WARNUNG:  DER RETURNCODE IST 4
// IF $RC = 8 THEN
* FEHLER:   DER RETURNCODE IST 8
/&
```

Massen-Insert im CICS für VSAM-Dateien

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 WRITE aus-gegeben 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.

Fehlermeldungen

Fehler können bei der Umwandlung oder bei der Ausführung eines Programms auftreten. Eine Diagnostik bei der Umwandlung erkennt bereits im Vorfeld den größten Teil der Fehlerursachen. Bei der Ausführung können Probleme auftreten, wenn Syntaxfehler nicht korrigiert wurden, wenn ein Newcopy auf eine aktive Programmlibrary ausgeführt wird oder bei allgemeinen Fehlern.

Allgemeine Fehler wie z.B. Data Check können genauso auftreten wie bei jedem CPG- oder HL1-Programm. Sie werden am einfachsten mit DEBUG lokalisiert.

Alle gemeldeten Fehler im QPG wurden behoben.

Syntaxfehler

Ein Programm wird zunächst durch einen Generator in Pseudo-Code übersetzt. Hierbei wird die Syntax nach formalen Fehlern überprüft.

Folgende Fehler werden erkannt:

01 Programm ist nicht verfügbar:

Library enthält nicht '*OA' oder '*BA' als User-Id. Dokument ist geschützt und der Protectioncode erlaubt keine Anzeige. Dokument ist gerade in Arbeit.

02 Programm Ausführungs-Pool ist voll:

Zu viele aktive Programme im Pool. Pool vergrößern.

03 Ungültiges Statement:

Allgemeine Fehlermeldung, z.B. Schreibfehler im Dokument.

04 DD Struktur fehlt:

Im Data Dictionary ist die gewünschte Struktur nicht vorhanden. Eventuell wird diese gerade im QTF bearbeitet.

05 Programm ist zu groß:

Zu viele Statements im Programm. Die maximale Größe des Programms wurde überschritten.

06 Feld doppelt definiert:

Ein Feld wird mehrfach in der Data Division definiert, oder es wird mit unterschiedlichen Längen in der Input Division verwendet.

07 Feld nicht definiert:

Für ein Feld wurde keine Länge angegeben. Bei Dateien wird CPGFRC nicht benutzt, bei HL1-Datasets ist CPGHIC nicht definiert oder bei Dateiausgabe mit Parameter VAR ist das Feld CPGVRL nicht definiert.

08 Prog Library nicht definiert:

Die Library ist nicht in der Directory QPGPXD enthalten. Sh. STAtus Command in der Servicetransaktion QPG.

09 DO/IF Syntax Fehler:

Zuviele DO-/IF-Stufen oder die Anzahl der END-Statements passt nicht zu den DO-/IF-Stufen. DO mit ENDIF oder IF mit ENDDO codiert.

10 Datei nicht definiert:

File-Anweisung ist nicht vorhanden.

11 Ein/Ausgabe fehlt:

FILE-, FIELD- oder SEGMENT-Anweisung fehlt in der Input oder Output Division.

12 Dateiart nicht unterstützt:

Es wurde eine Dateiart verwendet, die nicht zugelassen ist. Siehe Abschnitt Dateien.

13 Feld falsch definiert:

Die Feldlänge ist anders angegeben als erforderlich. Ein Feld hat die Länge 0, ein numerisches Feld ist länger als 15 Stellen oder ein Alphafeld ist länger als 256 Stellen.

Ein internes Feld wie UDATE oder CPGTID wurde in der Input-Division definiert.

Die Anzahl der Dezimalstellen ist größer als die Feldlänge.

14 Bereichsüberschreitung:

Bei Ein- oder Ausgabe befindet sich ein Feld oder eine Konstante außerhalb des zulässigen Bereichs. Z.B. ist die Satzlänge zu klein oder ein Feld in einem HL1-Datenkanal beginnt hinter Position 3936 und verletzt die HL1-Richtlinien. Der Fehler tritt auch auf, wenn Feldgruppen mit mehr als 10.000 Elementen definiert sind.

Fehlt in einer Ausgabe die zugehörige FILE- oder FIELD-Angabe, dann wird die Umwandlung mit dieser Fehlermeldung abgebrochen.

15 Label nicht/mehrfach definiert:

Ein Merkmal ist nicht oder mehrfach vorhanden.

16 EXHM Error:

Das Modul wurde nicht gefunden oder die HL1-Library für das Modul befindet sich nicht in der CICS-PPT.

17 SQL Error

Das Statement entspricht nicht den SQL-Regeln, oder Konstanten wurden nicht durch Leerstellen getrennt, oder die maximale Anzahl der Hostvariablen in einem Statement wurde überschritten.

18 Ungültige Operanden:

z.B. Alphafeld in Rechenoperation oder numerisches Feld ist bei EDIT oder SELCT angegeben.

19 Zu viele Operanden:

z.B. X = A + B + C

20 Ungültige Konstante:

Konstante ist falsch definiert oder Hex-Konstante enthält fehlerhafte Zeichen.

21 Ungültiges Füllwort:

Das Füllwort ist bei dieser Operation nicht erlaubt.

22 Ungültiger Service:

Es wurde ein falscher Wert angegeben.

23 Ungültiger Parameter:

Es wurde eine Konstante angegeben, wo z.B. ein Feld oder Dateiname erwartet wurde. HL1-Storage 'HS' wurde in der Eingabe für andere Dateiarten als HL1-Datasets angegeben. Bei einer CHAIN-Operationen für eine VSAM-KSDS-Datei ist entweder kein KEY angegeben oder die Länge des Schlüssels ist kleiner als die im DD angegebene KEY-Länge.

Wenn bei CHAIN, READ oder READB auf VSAM-KSDS-Dateien das interne Key-Feld CPGK.. benutzt wird, dann wird die Umwandlung mit einer Fehlermeldung abgebrochen.

24 Ungültiger Index:

Der Index ist nicht numerisch oder negativ. Bei Feldgruppen mit festem Index ist der Index 0 oder zu groß.

25 PWA ist zu groß:

Zu viele oder zu lange Felder definiert. Bei der Library oder beim Dokument die Erweiterte Adressierung einschalten.

26 Options fehlt:

Bei Dateiverarbeitung fehlt z.B. die OPTIONS DAT oder PWA. Bei HTMLI und HTMLO ist die OPTIONS HTML erforderlich.

27 xxxxxx ist reserviert.

Das Feld xxxxxx darf in dieser Operation nicht benutzt werden, z.B.: FILL ' ' to UDATE.

28 WHEN/EVALUATE Syntax Fehler

Eine EVALUATE-Gruppe wurde falsch codiert, z.B. WHEN ohne vorheriges EVALUATE-Statement, fehlendes END-EVALUATE Statement oder kein WHEN nach EVALUATE.

29 Eintrag fehlt:

Bei 'PROCEDURE DIVISION' wurden keine Rechenbestimmungen gefunden.

30 BEGSR/ENDSR Syntax Fehler:

Nach BEGSR fehlt ENDSR oder vor ENDSR fehlt BEGSR. U.U. wurde nach ENDSR ein anderes Statement als BEGSR codiert.

Das fehlerhafte Statement wird mit angezeigt.

Wird ein Fehler, der vom Generator erkannt wurde, nicht berichtigt, so erfolgt bei der Ausführung ein Abbruch mit dem Code 'ECPR'.

Bei Programm Check Interruptions wird im CICS die Testhilfe Debug aufgerufen, um den Fehler unmittelbar analysieren zu können, wenn nicht in der Anwendung mit der Operation DUMP ON ein Transaction DUMP aktiviert wird. Siehe Seite 9210. Im Batch wird bei einem Programmabbruch generell ein Speicherauszug erstellt.

Fehler bei Druckprogrammen

Bei Syntaxfehlern in Programmen, die auf einem Drucker gestartet werden, erfolgt die Ausgabe der Fehlermeldung auf dem Drucker und auf der Konsole. Die Ausgabe auf dem Drucker entspricht der Fehleranzeige bei Ablauf an einem Bildschirm. Die Transaktion wird mit dem Abend-Code 'ECPR' beendet.

Fehler bei None-Terminalprogrammen

Bei Syntaxfehlern in Hintergrundprogrammen im CICS ohne Terminal-Id oder bei Programmen, die in einer APPC-Verbindung ablaufen (z.B. Anwendungen mit NetPage im Intranet-/Internet-Umfeld), werden Fehlermeldungen auf der Konsole ausgegeben.

Fehler im Batch

Bei Batchprogrammen die z.B. aus QPGUTIL, HL1 oder RPG aufgerufen werden, erfolgt die Ausgabe der Fehlermeldung auf Konsole.

Tip zum Suchen von Syntaxfehlern:

Das Programm im QTF mit Auswahl 'X' (eXecute) und der Taste PF4 aufrufen. Hierbei erfolgt nur eine neue Umwandlung mit Syntaxprüfung aber ohne Ausführung. Die Syntaxfehler können dann am Bildschirm leicht korrigiert werden.

Fehler bei der Ausführung mit Code ECPR

Ein Abbruch mit diesem Code tritt auf, wenn ein Programm ausgeführt wird, das nicht fehlerfrei übersetzt wurde.

Vor einem Abbruch mit ECPR bei der Ausführung eines fehlerhaften Programms am Bildschirm, das nicht mit New-Copy korrigiert wurde, wird angezeigt, um welches Programm es sich handelt.

Handling bei einem Programmabbruch im CICS

Bei einem Programmabbruch im CICS innerhalb der QPG-Ausführung wird jetzt standardmäßig die Testhilfe DEBUG aufgerufen, wenn das bei der Installation so eingestellt wurde oder wenn mit DEBUG getestet wird. Das gilt auch dann, wenn der Debugger auf Ausgabe LOG-Daten umgeschaltet war.

Ein Fehler lässt sich so unmittelbar an der Stelle analysieren, an der er aufgetreten ist. Z.B. bei Abbrüchen infolge von Data Checks kann sofort kontrolliert werden, welche Datenfelder ungültige Inhalte haben. Mit der Operation DUMP und den Parametern 'ON' oder 'OFF' kann das Verhalten von QPG im CICS bei einem Programmabbruch innerhalb der QPG-Ausführung gesteuert werden.

Beispiel für die Anzeige nach einem Programmabbruch:

```
QPGDPD      Online Debug Facility          V.L  UID  TERM  tt.mm.jj  12.14UHR
-------------------------------------------------------------------------------
Programm       :   TPR        Library  :   TEST       Statement Nummer :     6
Transaction    :   QTF        Task-Nr. :   04124      Funktionstaste   :    DE
Modul          :   HPROG                              Restart PF-Key   :
-------------------------------------------------------------------------------
* F e h l e r *:   Programm-Abbruch
Bei  Operation :   EDIT                               File Return Code :

Faktor1        :              ,        :
Faktor2        :              ,        :
                                        ....+....1....+....2....+....3..
Ergebnisfeld   :   A10        ,        :
Service        :

Feld anzeigen  :              ,        :


                                        ....+....1....+....2....+....3..
-------------------------------------------------------------------------------
Enter: Weiter      PF4 : Trace Bedingungen PF8 : Feld anzeigen     Clear: Exit
PF1  : Hilfe       PF5 : Bildschirmdump    PF9 : Programm anzeigen
PF2  : User Bild   PF6 : Log Einzelsatz    PF10: Maske          anzg/dr:
```

Mit PF8 können jetzt die Feldinhalte angezeigt werden. Felder, die einen ungültigen Inhalt haben, werden in der Anzeige markiert (reversiv) dargestellt.

Folgende Fehler werden bei der Ausführung erkannt:

Programmabbruch	z.B. bei ungültigen Feldinhalten (Data Check)
Indexfehler	variabler Index ist zu hoch oder <= 0
Division	durch 0
Überlauf	bei Division oder COMPUTE Operation
CPGTCT	TCT User Area Length ist 0 oder zu klein

Maximalwerte

Folgende Maximalwerte können bzw. dürfen nicht überschritten werden:

Bereich	Maximalwert		Kommentar
Directory	1.000	Anzahl	Libraries für Test und Produktion.
Größe	2.000.000	Bytes	je Library.
Programm	32.767	Bytes	wenn die Library groß genug ist.
PWA	32.767	Bytes	inklusive Praefix und interne Felder.
CPGCOM	32.750	Bytes	Common-Area-Länge.
Dateien	32.767	Bytes	Satzlänge.
Dateien	256	Bytes	Keylänge bei VSAM KSDS.
Datenkanal	3.944	Bytes	bei HL1-Modulen
Storages	32.763	Bytes	Satzlänge für Temporary Storages.
QPCFXL	999	Bytes	Satzlänge für QPCF-Ausgabe.
Arrays	10.000	Anzahl	Elemente für jede Feldgruppe.
DB2	512	Anzahl	Hostvariablen je SQL-Statement.
Feldnamen	18	Stellen	in Verbindung mit DD, sonst 6 Stellen.
Labels	8	Stellen	Namen von Subroutines.