CPG5-RPG-Schnittstelle

Kapitel 2: Funktionen 2000

Die Schnittstelle stellt CPG-Funktionalitäten für RPG-Batch-Programme zur Verfügung, und zwar:

- **Data Dictionary**

  Datenstrukturen des CPG..Data Dictionary können in den I- und O-Karten angezogen werden. Das Coding ist das gleiche wie für CPG im CPG1-Format.

- **Externe HL1-Subroutines**

  HL1-Subroutines können genau wie im CPG im CPG1-Format mit EXHM eingebunden werden.

- **Externe QPG-Subroutines**

  QPG-Subroutines können mit PROG wie aus CPG-Programmen aufgerufen werden.

- **QPG-Datasets für Primary, Secondary und Output Dateien**

  QPG-Datasets können benutzt werden, um den Datenzugriff zu flexibilisieren. Um den physischen Dateizugriff aus dem RPG-Programm auszulagern, ist nur eine Änderung der F-Karte erforderlich. Siehe dazu Kapitel 3400.

  Solche Dataset-Module können für andere Dateitypen ebenfalls benutzt werden, und zwar mit der ‘PROG-Schnittstelle’. Siehe dazu Kapitel 3500.

- **Externer List-Output mit dem CPG-Befehl LIST**

  QLF – Quick List Facility ist im QTF-Handbuch im Kapitel 5000 beschrieben.

- **Vergleich von Datumsfeldern mit dem Befehl COMP in Verbindung mit Spalte 53**

  Siehe dazu Kapitel 3700.

Kapitel 3: Regeln 3000

Regeln für Data Dictionary im RPG 3100

Weitergehende Beschreibungen finden Sie im Kapitel 2600 des CPG2-Programmiererhandbuchs.

Die Spalten für die Grundfunktionen:

Data Dictionary in den I-Karten

```
Spalten   7-14      Dateiname
         21-22      DD (für Data Dictionary)
         23-24      Satzart
            42      T (für Textzeile mit Felddokumentation)
         53-60      Referenz-Datei
```

Data Dictionary in den O-Karten

```
Spalten   7-14      Dateiname
         19-20      DD (für Data Dictionary)
         21-22      Satzart
         45-52      Referenz-Datei
```

Regeln für den Einsatz von HL1-Modulen im RPG 3200

HL1-Module können im RPG genau wie im CPG eingesetzt werden.

Die Spalten für die Einträge sind:

```
Spalte       6      I  (Beschreibung des Datenkanals)
          7-12      Name des Datenkanals
         15-16      HS (für HL1 structure)

Spalte       6      C  (Aufruf des HL1-Moduls)
         28-32      EXHM (Befehl  EXecute HL1 Module )
         33-38      Name des HL1-Moduls
         43-48      Name des Datenkanals
```

HL1- Module im RPG – Hinweise und Einschränkungen 3250

HL1-Besonderheiten

- Felder im Datenkanal müssen entweder alphanumerisch oder gepackt numerisch beschrieben sein.

- Der Datenaustausch zwischen dem rufenden und dem gerufenen Programm findet aufgrund der Position im Datenkanal statt. Der Datenkanal ist im rufenden Programm in den I-Karten beschrieben, im gerufenen Programm in den (aller)ersten D-Karten.

- Beide Bereiche müssen absolut identisch sein, daher wird der Einsatz von Data Dictionary zur Beschreibung von Datenkanälen unbedingt empfohlen.

RPG-Besonderheiten

- Datenkanäle in RPG-Programmen werden intern als Datenstrukturen behandelt. Für RPG ist also der Datenkanal eine Datenstruktur. Weil Felder aber nicht in mehreren verschiedenen Datenstrukturen liegen dürfen, ist die Flexibilität hier geringer als im CPG.

  Die QPG-Schnittstelle, die im nächsten Kapitel beschrieben wird, bietet aber eine höhere Flexibilität.

Syntax für den Einsatz von QPG-Modulen im RPG 3300

In den C-Karten:

```
OperationsCode:   PROG
Faktor 2:         Name des Moduls (oder leer bei variablem PROG)
Ergebnisfeld:     (optional) QTF-Library, in der das Modul steht  oder
                  bei variablem PROG  ein 32-stelliges alphanumerisches Feld,
                  das die nötigen Informationen für den PROG-Befehl enthält
```

Die Logik des Datenaustauschs im QPG wird hier als bekannt vorausgesetzt.

Regeln für den Einsatz von QPG-Datasets im RPG 3400

Für primary, secondary und output files kann die Dateiverarbeitung einfach in ein QPG-Modul, ein sogenanntes Dataset-Modul, ausgelagert werden.

Die Syntax ist einfach - nur zwei Einträge in die F-Karte der Datei:

```
Spalte   7-12:   Name der logischen Datei(hier SAMPLE)
Spalten 31-32:   SP für Special PROGram
Spalten 40-43:   QTF-Library, in der das QPG-Dataset-Modul steht
```

Mit diesen Einträgen wird der physische Datenzugriff aus dem Programm ins Dataset-Modul ausgelagert. Der Vorteil ist, dass alle Änderungen am Dateizugriff in einem Modul vorgenommen werden können, das vom RPG-Batch-Programm genauso wie von CPG-Online- und Batchprogrammen aufgerufen werden kann.

Ein Beispiel für ein Dataset, um die Datei CPGWKV aus RPG-Programmen zu lesen und zu verändern:

```
options  define dataset.              * Dataset-Module SAMPLE in der Library TEST
file CPGWKV.                          * physische Datei definieren
-d.
    CPGIOA 1000 * 1.                  * Satzlänge von CPGWKV ist 1000 Bytes
    opcode  2.                        * Operationscode
-i.
    file CPGWKV.
      1  1000   CPGIOA
-c.
    opcode = cpgfrc.                  * Operationscode, der vom rufenden Programm
                                      * kommt, verschieben
    cpgfrc = '  '.                    * ReturnCode initialisieren
                                      *
    evaluate.                         * genau eine Alternative
      when opcode = 'R'.              * read
        read CPGWKV
      when opcode = 'O '.             * open
        open CPGWKV
      when opcode = 'OI'.             * open for input
        open CPGWKV input
      when opcode = 'OO'.             * open for output
        open CPGWKV output
      when opcode = 'OU'.             * open for updat
        open CPGWKV updat
      when opcode = 'OR'.             * open reuse
        open CPGWKV reuse
      when opcode = 'G '.             * chain (get)
        KEY chain CPGWKV
      when opcode = 'U'.              * updat
        updat CPGWKV
      when opcode = 'N'.              * write
        write CPGWKV
      when opcode = 'C'.              * close
        close CPGWKV
    end-evaluate
```

Ein solches Dataset kann von RPG-, CPG- und QPG-Programmen und -Modulen mit den CPG-Operationen für die Dateiverarbeitung angesprochen werden (wenn der entsprechende Eintrag in der F-Karte gemacht wurde).

Der Befehl für ein sequentielles READ wäre in diesem Fall READ SAMPLE. Üblicherweise benutzt man den gleichen Namen für die physische wie für die logische Datei. Hier im Beispiel heißt die logische Datei SAMPLE, die physische Datei ist CPGWKV.

Das rufende Programm setzt einen Operationscode in das Feld CPGFRC entsprechend dem programmierten Befehl (‘R ‘ für READ im Beispiel oben) und ruft das Dataset-Modul auf. **Die Daten werden ausgetauscht in einer Area, die so groß ist wie die vereinbarte Satzlänge der logischen Datei.**

In diesem Beispiel funktioniert der Datenaustausch, weil die Definition der I/O-Area in den ersten Stellen der Data Division identisch mit dem Datei-Input (in der Input Division) ist.

Weitere Regeln für QPG-Datasets im RPG 3500

Für andere Dateitypen als primary, secondary und output files kann die Dateiverarbeitung ebenfalls in ein QPG-Modul ausgelagert werden.

In einem solchen Fall wird das Module mit zwei Statements aus den C-Karten aufgerufen. Am Beispiel der Operation READ:

```
C                   MOVE 'R '       CPGFRC
C                   PROG SAMPLE     TEST
```

Beachte:

Datasets werden nicht automatisch eröffnet. Voraussetzung für die Verarbeitung ist ein explizites OPEN (hier: for input):

```
C                   MOVE 'OI'       CPGFRC
C                   PROG SAMPLE     TEST
```

Der Operationscode wird in das zweistellige alphanumerische Feld CPGFRC gesetzt. Die Inhalte von CPGFRC sind im Kapitel 3400 beschrieben. Das Dataset-Modul wird mit der Operation PROG aufgerufen.

In diesem Fall geschieht der Datenaustausch automatisch bei Feldern mit gleichem Namen und gleichen Feldeigenschaften (im RPG-Programm wie im QPG-Modul).

Vergleiche von Datumsfeldern im RPG 3700

CPG kennt den direkten Vergleich von Datumsfeldern der verschiedensten Formate.

Diese Vergleichoperationen sind im CPG2-Programmiererhandbuch beschrieben unter IF-DAT (für das Format ttmmjj), IF-DATI (für das Format jjmmtt) und IF-DATK (für das Format jjttt).

Dies Funktionalität kann auch mit COMP im RPG genutzt werden, wenn man die folgenden Einträge in Spalte 53 der Operation benutzt:

D für ttmmjj-Formate
I für jjmmtt-Formate
K für jjttt-Formate

Kapitel 4: Hints and Tips 4000

Optimierung der Ausführungszeit 4010

Um **LIST-**Output (signifikant) zu optimieren:

Die Operation LIST verarbeitet im Normalfall das gesamte LIST-Dokument. Insbesondere, wenn Ausgabelogik im LIST-Dokument programmiert ist, können LIST-Dokumente eine stattliche Größe bekommen. Um zu vermeiden, dass immer alle Statements ausgeführt werden, gibt es die Steuerbefehle **§endlist** und **§endsect;** sie beenden die LIST-Verarbeitung explizit. (Siehe QTF-Handbuch für detailliertere Beschreibung)

Im Zusammenhang mit dem Einsatz des Steuerbefehls §endlist sollten die Sections eines LIST-Dokuments nach der Häufigkeit der Benutzung **sortiert** werden. Normalerweise ist die Detail-Zeile die am häufigsten ausgeführte; sie sollte deshalb an den Kopf des Dokuments sortiert und mit §endlist abgeschlossen werden.

Um **Dataset-Module** für Batchverarbeitung (ein wenig) zu optimieren:

Normalerweise haben alle Datasets den gleichen Aufbau: In einer EVALUATE-Gruppe werden alle vorkommenden Operationscodes für Dateiverarbeitung abgefragt und programmiert. Eine EVALUATE-Gruppe führt höchstens eine Alternative aus. Wenn eine Alternative ausgeführt wird, verzweigt die Operation intern zum END-EVALUATE-Statement.

Beim Verarbeiten von Massendaten im Batch führt eine Sortierung nach Häufigkeit einer Operation zu einer Verringerung der Rechenzeit: Normalerweise ist das READ die häufigste Operation und sollte deshalb die erste Abfrage in der EVALUATE-Gruppe sein. OPEN und CLOSE als Gegenbeispiel werden in der Regel nur ein Mal pro Programm ausgeführt.

Kapitel 5: Interne Felder und Schalter 5000

CPGFRC File Return Code und Interface Control, 2 Stellen alphanumerisch 5010

CPGFRC wird für zwei Aufgabe verwendet:

- Auf dem Weg vom rufenden Programm zum Dataset enthält CPGFRC die Information über die Operation, die im Dataset ausgeführt werden soll:

```
'B' READB
'C' CLOSE
'D' RNDOM
'E' EXCPT
'G' CHAIN
'GC' for CHAIN CHECK
'GP' for CHAIN CHECK for Update
'GU' for CHAIN for UPDATE
'I' READI
'L' DELET       ( siehe Einschränkungen in Kapitel 7000 ! )
'N' WRITE
'O' OPEN
'OI' for OPEN for Input
'OO' for OPEN for Output
'OR' for OPEN Reuse
'OU' for OPEN for Update
'R' READ
'S' SETLL
'U' UPDAT
'Z' CHECK
```

- Auf dem Weg zurück vom Dataset zum rufenden Programm enthält CPGFRC den Wert des File Return Codes. Dieser Wert muss ‘manuell’ im rufenden RPG-Programm in den entsprechenden Schalter umgesetzt werden:

DK	Duplicate Key nach READ einer AIX-Datei
DR	Duplicate Record nach Hinzufügen
EF	End of File nach READ, READ-BACK und SETLL
NC	Not closed nach CLOSE
NF	Not found nach CHAIN oder CHECK, OPEN, CLOSE
NO	Not open nach OPEN oder CHECK
Blank	wenn keine Besonderheit aufgetreten ist

CPGLCT Line Counter, 5 Stellen numerisch 5050

Der Overflow-Schalter OF wird nicht automatisch zwischen RPG-Programm und LIST-Dokument ausgetauscht. Die Overflow-Logik im RPG-Programm muss ersetzt werden durch eine Abfrage des Wertes des internen Zeilenzählers CPGLCT .

Der **Vorteil**: Die gesamte LIST-Output-Steuerung wird aus dem Programm ausgelagert. Wie Abfragen von Feldinhalten beim LIST funktionieren, steht im QTF-Handbuch.

CPGPCT Page Counter, 5 Stellen numerisch 5070

Das RPG-interne Feld PAGE wird nicht automatisch zwischen RPG-Programm und LIST-Dokument ausgetauscht.

Es gibt zwei Möglichkeiten, dieses 'Problem' zu lösen: Entweder man definiert das Feld PAGE explizit im RPG-Programm oder man ersetzt es durch das CPG-interne Feld CPGPCT, das im LIST-Dokument automatisch mit der Funktion :p erhöht werden kann. (Details wiederum im QLF-Teil des QTF-Handbuchs!)

OF Overflow-Schalter 5100

Der Overflow-Schalter OF wird nicht automatisch zwischen RPG-Programm und LIST-Dokument ausgetauscht. Die Overflow-Logik im RPG-Programm muss ersetzt werden durch eine Abfrage des Wertes des internen Zeilenzählers CPGLCT .

Der **Vorteil**: Die gesamte LIST-Output-Steuerung wird aus dem Programm ausgelagert. Wie Abfragen von Feldinhalten beim LIST funktionieren, steht im QTF-Handbuch.

Schalter 5200

Schalter werden nicht automatisch zwischen RPG-Programm und LIST-Dokumenten oder QPG-Modulen ausgetauscht. Obwohl das LIST-Facility die Abfrage von Schaltern kennt, **können RPG-Schalter nicht in LIST-Dokumenten abgefragt werden.**

Falls nötig, muss man die Schalter im Programm in Feldinhalte umsetzen, die dann mit §if im LIST-Dokument abgefragt werden können.

Section 7 Einschränkungen 7000

Für **HL1-Module** und **HL1 Datenkanäle**

- Felder im Datenkanal müssen alphanumerisch oder gepackt numerisch sein.

- Datenkanäle werden in RPG-Programmen intern als Datenstrukturen behandelt. Für RPG ist der Datenkanal eine Datenstruktur, und Felder können nicht Teil verschiedener Datenstrukturen sein. Das muss beachtet werden, wenn mehrere HL1-Module von einem RPG-Programm aufgerufen werden.

Für die Operation **LIST**

- Die Abfrage von Schaltern des RPG-Programms ist im LIST-Dokument nicht möglich.

  –> Stattdessen können Feldinhalte abgefragt werden (mit §if im Steuerbefehl §section)

Für **QPG-Datasets** sind die folgenden Verarbeitungsarten nicht unterstützt:

- Operation READP (READ-PAGE) –> Stattdessen bitte eine Schleife programmieren !

- Die Operation RNDOM*ALL arbeitet nicht für die QPG-Datasets –> Separate RNDOM geben !

- **DEL** (fürs Löschen) in den **O-Karten** ist nicht automatisch unterstützt.

  Statt des Eintrags DEL trägt man Blanks in die entsprechenden Spalten der Output-Satzbestimmung ein. Zusätzlich muss stattdessen eine Output-Feldbestimmung eingefügt werden mit einer Konstanten, die in dem ‚Update-Teil' des Datasets abgefragt werden muss. (DEL wird wie ein Sonderfall des Update behandelt).

  Eine zweite Möglichkeit ist der Austausch des EXCPT im RPG-Programm durch einen expliziten Aufruf des Datasets mit PROG, wobei vorab das Feld CPGRFC auf ‘L ‘ wie Delete gesetzt wird.

Beispiel 1: LIST statt O-Karten im RPG 8000

Original-Programm:

```
H              J                                                      ORIGIN
FCPGKSD  IP  V     500            KSDS
FPRINTER O   F     132     OF     PRINTERSYSLST

ICPGKSD  NS  01
I                                       1   5 KDNRA
I                                      21  50 FIRMA
I                                      71  75 PLZ
I                                      76  95 ORT
I                                      96 120 STR

C   01      ANZ       ADD  1          ANZ     50

OPRINTER D  101   1P
O       OR        OF
O                                       5 'KD-NR'
O                                      11 'FIRMA'
O                                      46 'PLZ   ORT'
O                                      67 'STR'
O                                      91 'SEITE'
O                          PAGE  Z     96
OPRINTER D  2     1P
O       OR        OF
O                                      24 '------------------------'
O                                      48 '------------------------'
O                                      72 '------------------------'
O                                      96 '------------------------'
OPRINTER D  1     01
O                          KDNRA       5
O                          FIRMA      36
O                          PLZ        42
O                          ORT        63
O                          STR        89
OPRINTER T 11     LR
O                                      24 '------------------------'
O                                      48 '------------------------'
O                                      72 '------------------------'
O                                      96 '------------------------'
OPRINTER T  1     LR
O                                      14 'ANZAHL SAETZE:'
O                          ANZ    Z    20
```

Neues Programm:

```
     H              J                                                    EX8000

     FCPGKSD  IP  V     500            KSDS

     ICPGKSD  KF  01
     I                                       1   5 KDNRA
     I                                      21  50 FIRMA
     I                                      71  75 PLZ
     I                                      76  95 ORT
     I                                      96 120 STR

     C   01       ANZ        ADD  1         ANZ     50
     C   01                  LIST CPGKSDR   DETAIL
     CLR                     LIST CPGKSDR   TRAILR
```

Mit dem LIST-Dokument:

```
Document CPGKSDR  LIST Liste cpgksd                   3 page  1    line  1     width  72      2.4
....+....1....+....2....+....3....+....4....+....5....+....6....+....7.<..+....8....+....9....+..100
§SECTION DETAIL    §IF CPGPCT = 0
§PERFORM HEADER
§SECTION DETAIL    §IF CPGLCT > 60
§PERFORM HEADER
§SECTION DETAIL
§NAMETAB KDNRA
%     §FIRMA                       §PLZ  §ORT                §STR
§ENDLIST
§SECTION HEADER
§NEWPAGE
§NAMETAB CPGPCT:PRZ
KD-NR FIRMA                        PLZ   ORT                 STR                   SEITE    %
-------------------------------------------------------------------------------------------------

§ENDSECT
§SECTION TRAILR

-------------------------------------------------------------------------------------------------
§NAMETAB ANZ:RZ
ANZAHL SAETZE:     %
**** end ****


....+....1....+....2....+....3....+....4....+....5....+....6....+....7.<..+....8....+....9....+..100
End of document.
```

Beispiel 2a: 1:1 QPG-Dataset im RPG-Programm 8100

Original-Programm:

```
     H               J                                                  EX8000

     FCPGKSD  IP  V     500            KSDS

     ICPGKSD  KF  01
     I                                       1   5 KDNRA
     I                                      21  50 FIRMA
     I                                      71  75 PLZ
     I                                      76  95 ORT
     I                                      96 120 STR

     C   01      ANZ       ADD  1         ANZ     50
     C   01                LIST CPGKSDR   DETAIL
     CLR                   LIST CPGKSDR   TRAILR
```

Neues Programm:

```
     H               J                                                  EX8100

     FCPGKSD  IP  V     500   SP       TEST

     ICPGKSD  KF  01
     I                                       1   5 KDNRA
     I                                      21  50 FIRMA
     I                                      71  75 PLZ
     I                                      76  95 ORT
     I                                      96 120 STR

     C   01      ANZ       ADD  1         ANZ     50
     C   01                LIST CPGKSDR   DETAIL
     CLR                   LIST CPGKSDR   TRAILR
```

Mit dem QPG-Dataset

```
options  define dataset.             * Das ist das Datasetmodul CPGKSD in TEST
file CPGKSD dd type PH.              * Physische Datei definieren
-d.
    cpgioa 500 * 1.                  * I/O-Area (Länge = Satzlänge der Datei)
    cpgfrc  2.                       * Interface Control Field vor CPGEDS !
    cpgeds 0 * 1. *------------------* End of Data Exchange section
    opcode  2.                       * Operationscode
-i.
    file CPGKSD.                     * physische Datei
      1  500  cpgioa
-c.
    opcode = cpgfrc.                 * Operationscode aus dem rufenden Programm
                                     * nach opcode verschieben
    cpgfrc = '  '.                   * Returncode initialisieren
    evaluate.                        * genau eine Alternative
      when opcode = 'R'.             * read
        read CPGKSD
      when opcode = 'O'.             * open for input
        open CPGKSD input
      when opcode = 'C'.             * close
        close CPGKSD
    end-evaluate
```

Anmerkungen:

Das ist ein kleines Beispiel, um die Technik zu erklären.

- Im RPG-Programm wird die F-Karte erweitert um die Einträge SP und QTF-Library TEST (in der das Dataset steht) statt der 'physischen' Einheit KSDS

- Das Dataset liest die physische Datei. Eine Beschreibung dieser Datei (zumindest QDDF für die Dateieigenschaften) wird im Data Dictionary benötigt. In den folgenden Beispielen hat die logische Datei den gleichen Namen wie die physische Datei. Die logische und die physische Datei können unterschiedlich aufgebaut sein. Deshalb sind sie in den Beispielen dieses Handbuchs als unterschiedliche Satzarten der Datei CPGKSD beschrieben. Die logische Datei ist die ohne Satzart, die physische Datei ist die mit der Satzart PH.

```
File Maintenance                          2.4   UID  TERM  tt.mm.jj   s.mmUhr
------------------------------------------------------------------------------

File name ............. CPGKSD   PH     Input/output type ..... U
Record format ......... V               Block length ..........
Record length ......... 500             Key length ............ 20
Addressing mode ....... K               File organization ..... V
Key position ..........    1            Device / Library ...... KSDS
Add ...................


Input/output mode .....                 Type of processing ....
SYSnnn ................                 Default specification..

Description ........... Demo für das Handbuch
Programmer ............ UID              Key type ..............
Remark .................                Prot-Code .............
Reference file .........                Reference record type .
Sort order .............                Compiler Prefix .......
Directory Field Check . N               QWS enabling ..........
                                        Structure protection ..

------------------------------------------------------------------------------
CPGKSD    change    Press ENTER to continue
```

- Die Daten werden ausgetauscht in einer Area, die so lang ist wie die physische Datei CPGKSD. Weil der Bereich größer ist als die maximale Feldlänge (256 bytes) im CPG, ist die Area hier als Feldgruppe definiert. Die I/O-Area wird immer in den (aller)ersten Stellen der Data Division definiert.

- Das Feld CPGEDS ist ein logisches Feld ohne wirkliche Länge. Üblicherweise wird es definiert wie oben angegeben. QPG-Module holen Feldinhalte vom rufenden Programm über den Feldnamen: Wenn Feldnamen und Feldeigenschaften im rufenden und gerufenen Modul gleich sind, findet ein Datenaustausch statt. CPGEDS markiert das Ende dieses Datenübergabebereichs (zum Beispiel, um Arbeitsfelder nicht zu übertragen). Alle Felder, die im gerufenen Modul hinter CPGEDS definiert sind, nehmen am Datenaustausch nicht teil.

- Das Dataset erhält die Information, welche Dateioperation ausgeführt werden soll, automatisch im Feld CPGFRC. Andererseits liefert CPGFRC automatisch das Ergebnis der Dateioperation zurück ans rufende Programm. Deshalb wird CPGFRC am Programmanfang in das Arbeitsfeld OPCODE geschoben und CPGFRC wird initialisiert.

- In diesem Beispiel sind nur die drei Funktionen OPEN, READ und CLOSE im Dataset programmiert. Es sind die Operationen, die im Beispiel benötigt werden. Normalerweise wird das Dataset alle Dateioperationen abdecken.

Beispiel 2b: Einsatz von Data Dictionary 8150

Da die Dataset-Technik der hier beschriebenen Schnittstelle die Daten in einer I/O-Area austauscht, ist der Einsatz von Data Dictionary nicht wirklich notwendig.

Der Vollständigkeit halber ist der Einsatz aber in diesem Kapitel beschrieben.

Existierende Strukturen (z. B. in RPG-I-Karten-Syntax) können mit einem Lattwein-Batch-Utility ins Data Dictionary geladen werden. Manuelles Anlegen und Pflegen von Strukturen ist natürlich ebenso möglich. Die Input-Struktur aus Beispiel 8100, ins Data Dictionary importiert, würde wie folgt aussehen:

```
Field Name from     to Byte P  Ae   Lng
-----------------------------------------
KDNRA          1     5     5           5
               6    20    15
FIRMA         21    50    30          30
              51    70    20
PLZ           71    75     5           5
ORT           76    95    20          20
STR           96   120    25          25
-----------------------------------------
```

Die Auswirkung auf die Codierung:

Original-Programm:

```
     H               J                                                  EX8100

     FCPGKSD  IP  V     500   SP       TEST

     ICPGKSD  KF  01
     I                                      1   5 KDNRA
     I                                     21  50 FIRMA
     I                                     71  75 PLZ
     I                                     76  95 ORT
     I                                     96 120 STR

     C   01      ANZ       ADD  1         ANZ     50
     C   01                LIST CPGKSDR   DETAIL
     CLR                   LIST CPGKSDR   TRAILR
```

Neues Programm:

```
     H               J                                                  EX8150

     FCPGKSD  IP  V     500   SP       TEST

     ICPGKSD  KF  01DD

     C   01      ANZ       ADD  1         ANZ     50
     C   01                LIST CPGKSDR   DETAIL
     CLR                   LIST CPGKSDR   TRAILR
```

Data Dictionary-Strukturen werden zur Umwandlungszeit vom Programm angezogen.

Beispiel 3a: QPG-Dataset mit erweiterter Logik (mit DD) 8200

In diesem Beispiel ist die Logik des Datasets um einen zusätzlichen Dateizugriff erweitert, der mehr Daten liefert als der bestehende READ-Befehl.

Wenn das RPG-Programm für diese Erweiterung unverändert bleiben soll, brauchen die zusätzlichen Felder nur ins Data Dictionary aufgenommen zu werden. Im Beispiel wurde DATCPG an die existierende Struktur angehängt:

Field Name	from	to	Byte	P	Ae	Lng
KDNRA	1	5	5			5
	6	20	15			
FIRMA	21	50	30			30
	51	70	20			
PLZ	71	75	5			5
ORT	76	95	20			20
STR	96	120	25			25
	121	500	380			
DATCPG	**501**	**506**	**6**			**6**

Die Dateibeschreibung muss dann entsprechend angepasst werden:

```
File Maintenance                              2.4   UID  TERM  dd.mm.yy   h.mmhrs
------------------------------------------------------------------------------

File name ............. CPGKSD          Input/output type ..... U
Record format ......... V               Block length ..........
Record length ......... 506             Key length ............
Addressing mode ....... S               File organization ..... P
Key position ..........                 Device / Library ...... TEST
Add ...................


Input/output mode .....                 Type of processing ....
SYSnnn ................                 Default specification..

Description ........... Demo für logische Struktur
Programmer ............ UID             Key type ..............
Remark .................                Prot-Code .............
Reference file ........                 Reference record type .
Sort order ............                 Compiler Prefix .......
Directory Field Check . N               QWS enabling ..........
                                        Structure protection ..

------------------------------------------------------------------------------
CPGKSD    change    Press ENTER to continue
```

Damit braucht die RPG-Anwendung nicht verändert, sondern nur neu kompiliert zu werden, um das zusätzliche Feld nutzen zu können.

Das Dataset mit dem zusätzlichen Dateizugriff könnte wie folgt aussehen:

```
options  define dataset.              * Das ist das Dataset Modul CPGKSD in TEST
file CPGKSD dd type PH.               * physische Datei definieren
file CPGKDN input.                    * Anmerkung 1
-d.
    cpgioa 500 * 1.                   * I/O-Area (Länge = Satzlänge der Datei)
    org cpgioa                        * Redefinition  Anmerkung 2
       kdnra  5.                      * erste fünf Stellen = Kundennummer
       org.                           * Ende der Redefinition
    datcpg  6.                        * zusätzliches Feld  Anmerkung 3
    cpgfrc  2.                        * Interface Control Field vor CPGEDS
    cpgeds 0 * 1. *-----------------* End of Data Exchange section
    opcode  2.                        * Operationscode
-i.
    file CPGKSD.                      * physische Datei
      1  500  cpgioa
    file CPGKDN.                      * andere physische Datei  Anmerkung 1
     172 177 datcpg
-c.
    opcode = cpgfrc.                  * Operationscode umspeichern, der vom
                                      * rufenden Programm kommt
    cpgfrc = '  '.                    * Returncode initialisieren
    evaluate.                         * genau eine Alternative
      when opcode = 'R'.              * read
        read CPGKSD
        kdnra chain cpgkdn.           * Get  Anmerkung 1
        if cpgfrc = 'NF'.             * File Return Code = Not Found ?
          datcpg = '      '.          * gefunden => datcpg gefüllt, sonst leer
        endif
      when opcode = 'O'.              * open for input
        open CPGKSD input
        open CPGKDN input.            * Anmerkung 4
      when opcode = 'C'.              * close
        close CPGKSD
        close CPGKDN
    end-evaluate
```

Anmerkungen:

1. Die zusätzliche Datei CPGKDN wird in der Files Division definiert, ihr Input (das Feld DATCPG) in der Input Division und der Zugriff in der Procedure Division.

2. Die Redefinition von CPGIOA beschreibt, dass die ersten 5 Stellen der I/O-Area den Wert der Kundennummer KDNRA enthalten. KDNRA ist der Schlüssel für den Direktzugriff (CHAIN) auf die Datei CPGKDN.

   Wenn dieses Feld irgendwo in der Mitte der I/O-Area läge, könnte es z. B. mit dem MOVEA-Befehl wie folgt ‚ausgelesen' werden: MOVEA CPGIOA(X) TO KDNRA.

3. Das 'neue' Feld CPGDAT ist im Data Dictionary in den Stellen 501 bis 506 beschrieben. Deshalb wird es hier direkt hinter die I/O-Area CPGIOA gelegt: Die Länge der I/O-Area wird von 500 auf 506 vergrößert !

4. OPEN und CLOSE für die zusätzliche Datei können in den entsprechenden Programmzweigen des EVALUATE-Befehls gemacht werden.

Beispiel 4: QPG-Dataset für 'andere' Dateitypen 8300

Für Dateien, die nicht 'primary', 'secondary' oder 'output' sind, **arbeitet die Dataset-Technik anders !** Das Dataset-Modul wird dann nicht automatisch über die Schnittstelle aufgerufen, sondern explizit mit PROG aus dem Programm. Das Feld CPGFRC muss dabei mit dem richtigen Wert gefüllt werden (siehe Kapitel 5010).

Diese Technik ist wesentlich flexibler, weil die Datenübergabe des QPG (by field name) zur Anwendung kommt. Andererseits sind natürlich mehr Eingriffe in den RPG-Code erforderlich. Das Dataset SAMPLE :

```
options  define dataset.              * Das ist das Datasetmodul SAMPLE in TEST
file CPGWKV.                          * physische Datei definieren
    define CPGWKV.                    * Felder definieren
    opcode  2.                        * operation code
-i.
    file CPGWKV dd type 41.           * physische Datei, Satzart 41
-c.
    opcode = cpgfrc.                  * Operationscode umspeichern, der vom
                                      * rufenden Programm kommt
    cpgfrc = '  '.                    * Returncode initialisieren
    evaluate.                         * genau eine Alternative
      when opcode = 'R'.              * read
        read CPGWKV
      when opcode = 'Z'.              * check
        check CPGWKV
      when opcode = 'O '.             * open
        open CPGWKV
      when opcode = 'OI'.             * open for input
        open CPGWKV input
      when opcode = 'OO'.             * open for output
        open CPGWKV output
      when opcode = 'OU'.             * open for updat
        open CPGWKV updat
      when opcode = 'OR'.             * open reuse
        open CPGWKV reuse
      when opcode = 'G '.             * chain (get)
        KEY chain CPGWKV
      when opcode = 'GC'.             * chain check
        KEY chain CPGWKV check
      when opcode = 'GP'.             * chain check for update
        KEY chain CPGWKV prot
      when opcode = 'GU'.             * chain for update
        KEY chain CPGWKV upd
      when opcode = 'S'.              * setll
        KEY setll CPGWKV
      when opcode = 'B'.              * readb
        readb CPGWKV
      when opcode = 'I'.              * readi
        readi CPGWKV 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
```

kann wie folgt aufgerufen werden:

```
H               J                                                    EX8300

F*   Keine F-Karte erforderlich

ISAMPLE      DSDD

C*
:
C                   MOVE `R `       CPGFRC  2
C                   PROG SAMPLE     TEST
:
C                   MOVE `N `       CPGFRC
C                   PROG SAMPLE     TEST
```

Anmerkungen:

1. Die Dateiverarbeitung ist jetzt aus dem Programm ausgelagert. Es wird also auch keine F-Karte benötigt.

2. Der File Input kann durch die Beschreibung einer Datenstruktur ersetzt werden (Eintrag DS in Spalten 19 und 20). Die Datenstruktur für ein 1:1-Dataset is die gleiche wie für die physische Datei. Das bedeutet, dass nur eine einzige Data Dictionary-Struktur benötigt wird. Data Dictionary wird zur Umwandlungszeit angezogen, wenn in den Spalten 21 und 22 DD eingetragen ist. Wenn eine Satzart benutzt würde, würde sie in die Spalten 23 und 24 eingetragen.

3. Das Dataset wird mit PROG aufgerufen, der Name des Datasets steht dabei in Faktor 2 und die QTF-Library im Ergebnisfeld. Vor dem PROG muss CPGFRC definiert und mit dem ‘passenden’ Wert für die dateiverarbeitende Operation (siehe Kapitel 5010) gefüllt werden.

Beispiel: QPG-Dataset mit I/O-Area – vollständiges Beispiel 8400

```
options  define dataset.              * Das ist das Datasetmodul SAMPLE in TEST
file MYFILE.                          * physische Datei definieren
-d.
    define IOAREA.                    * IOAREA, im Data Dictionary beschrieben
    cpgfrc  2.                        * Interface Control Field CPGFRC
    cpgkey  64.                       * enthält Faktor 1 der Dateioperation
    CPGEDS  0 * 1.  *-----------------* End of Data Exchange Area
    opcode  2.                        * Operationscode
-i.
    file MYFILE dd ref IOAREA.        * physische Datei (mit dem Feld KEY)
-c.
    opcode = cpgfrc.                  * Operationscode verschieben
    key = cpgkey.                     * (siehe Anmerkungen)
    cpgfrc = '  '.                    * Returncode initialisieren
    evaluate.                         * genau eine Alternative
      when opcode = 'R'.              * read
        KEY read MYFILE
      when opcode = 'G '.             * chain (get)
        KEY chain MYFILE
      when opcode = 'GC'.             * chain check
        KEY chain MYFILE.             * ohne Servicefunktion CHECK
      when opcode = 'GP'.             * chain check for update
        KEY chain MYFILE upd.         * ohne Servicefunktion CHECK
      when opcode = 'GU'.             * chain for update
        KEY chain MYFILE upd
      when opcode = 'S'.              * setll
        KEY setll MYFILE
      when opcode = 'B'.              * readb
        readb MYFILE
      when opcode = 'D'.              * rndom
        rndom MYFILE
      when opcode = 'U'.              * updat
        updat MYFILE
      when opcode = 'N'.              * write
        write MYFILE
      when opcode = 'L'.              * delet
        KEY delet MYFILE
      when opcode = 'C'.              * close
        close MYFILE
      when opcode = 'Z'.              * check
        check MYFILE
      when opcode = 'O'.              * open
        fill x'00' cpgkey
        evaluate
          when opcode = 'O '.         * open
            open MYFILE
          when opcode = 'OI'.         * open for input
            open CPGWKV input
          when opcode = 'OO'.         * open for output
            open CPGWKV output
          when opcode = 'OU'.         * open for updat
            open CPGWKV updat
          when opcode = 'OR'.         * open reuse
            open CPGWKV reuse
        end-evaluate
    end-evaluate
    cpgkey = KEY
```

Anmerkungen:

In the Beispielen oben ist der Einsatz von QPG-Datasets für den RPG-Kenner beschrieben. Das folgende Beispiel zeigt Erweiterungen für den Einsatz aus CPG-Programmen (für die Datasets, die Daten über eine I/O-Area austauschen):

Data Division

- In den allerersten Stellen muss die gesamte I/O-Area definiert werden.

- CPGFRC sollte ebenfalls am Anfang der Data Division definiert werden – auf jeden Fall vor der Definition von CPGEDS (siehe unten). CPGFRC bringt den File Return Code zurück zum rufenden Programm – wäre CPGFRC hinter CPGEDS definiert, würde kein Datenaustausch stattfinden!

- Das Feld **CPGKEY** liefert den Faktor 1 der Dateioperation. Wenn andere Schlüsselfelder als das im Dataset definierte Schlüsselfeld im CPG-Programm zur Anwendung kommen, ist dies der einzig mögliche Weg, den Wert dieses Schlüsselfeldes ins Dataset zu bekommen. **Deshalb muss auch CPGKEY vor CPGEDS definiert werden.**

- CPGEDS markiert das Ende des Bereichs, der zwischen Anwendung und Dataset ausgetauscht wird. Durch die Definition von CPGEDS vermeidet man, dass Arbeitsfelder des Datasets in die Anwendung übernommen werden und dort Werte überschreiben. Das ist insbesondere wichtig, wenn Anwendungen mit einer Menge unterschiedlicher Feldnamen das Dataset nutzen oder wenn das Dataset eine erweiterte Logik hat. **Arbeitsfelder werden hinter CPGEDS definiert**.

Input Division

- Die Feldbestimmungen für die physische Datei müssen mit der IOAREA übereinstimmen.

Procedure Division

- Das Feld CPGKEY wird dem Schlüsselfeld der Dateioperation am Anfang des Datasets zugewiesen (und wird am Ende des Datasets daraus zurückgeladen). Um die richtige Verarbeitung auch bei gepackten und generischen Schlüsseln sicherzustellen, wird **CPGKEY im OPEN-Zweig** auf hexadezimal '00' **initialisiert.**

- **CHAIN**:

  Um sicherzustellen, dass die Dateidaten zur Zeit eines UPDAT oder WRITE immer vollständig sind – sogar wenn die Struktur in der Anwendung unvollständig ist – empfehlen wir, **bei einem CHAIN-Befehl immer die gesamte Struktur** zu **holen**, auch wenn im Programm die Servicefunktionen 'C' für CHAIN CHECK oder 'P' für CHAIN CHECK FOR UPDATE benutzt werden.

  CHAIN füllt dann die gesamte File Work Area im Dataset. Die Services 'C' und 'P' verhindern, dass die Daten zur Anwendung übertragen werden. Wenn der nächste Aufruf des Datasets ein UPDATE oder WRITE ist, ist die File Work Area schon gefüllt – die geänderten Feldinhalte werden dann von der Anwendung geschickt.