PEARL

ASSOCIATION
VEREIN E.V.

Axel Kappatsch

Überblick über die Echtzeitprogrammiersprache PEARL

Kernforschungszentrale Karlsruhe
PDV-Berichte; KfK-PDV 140; September 1977

I N H A L T :

I. EINLEITUNG

PEARL (Process and Experiment Automation Realtime Language) wurde in Deutschland seit 1969 als höhere Sprache für die Programmierung von Echtzeitaufgaben entwickelt. Darunter sind in erster Linie Steuerungen von Prozessen jeder Art zu verstehen, z.B.

- Kommunikationsprozesse in Informationssystemen
- Experimentsteuerungen im wissenschaftlichen Bereich
- Abwicklung und Überwachung von industriellen Fertigungsprozessen etc.

Die besonderen Anforderungen an eine Sprache zur Prozeßsteuerung betreffen daher die

- Abwicklung zeitlich paralleler Teilvorgänge

und die

- Anpassung an ein breites Spektrum von Peripheriegeräten.

Die Schwerpunkte der PEARL-Sprachentwicklung lagen demgemäß im Echtzeit- und E/A-Bereich. Diese wurden in den durch die algorithmischen Sprachkonzepte definierten Rahmen eingefügt.

Im folgenden soll ein informeller Überblick über die wesentlichen Elemente von PEARL gegeben werden; bezüglich Einzelheiten sei auf den PDV-Bericht 130 /1/ verwiesen. Der Standard-Subset von PEARL, Basis-PEARL genannt, ist entsprechend in /2/ beschrieben. Ein historischer Abriß der PEARL-Entwicklung findet sich in /3/.

II. PROGRAMMSTRUKTUR

1. MODULEs

PEARL-Programme bestehen aus unabhängig übersetzbaren Einheiten, sog. MODULEs. Verbindungen zwischen MODULEs werden über GLOBALe Größen hergestellt (Abb. 1).

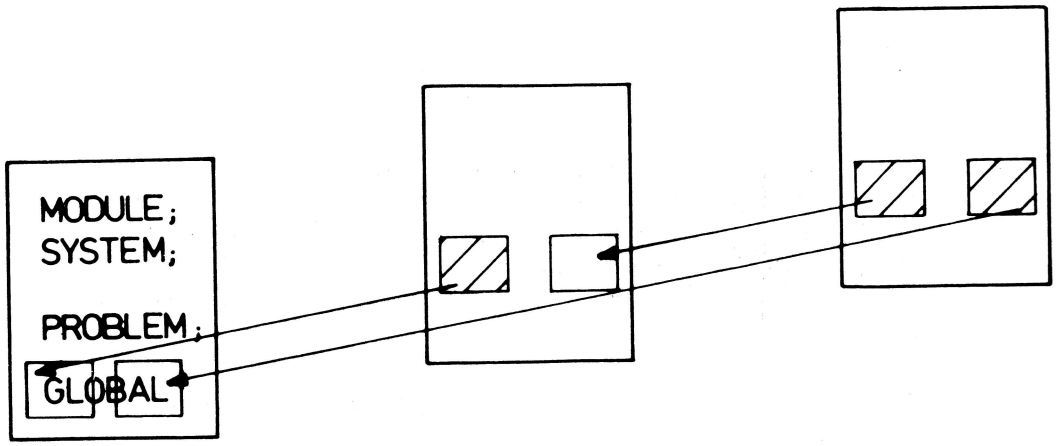


Abb. 1: MODULEs und ihre Verbindung über GLOBALe Größen (□). Schraffierung markiert die Ablagestelle.

Ein MODULE besteht i.a. aus einem Teil, der die Systemumgebung beschreibt (SYSTEM-division) und der eigentlichen Formulierung des Problems (PROBLEM-division).

2. SYSTEM-Teil (SYSTEM-division)

Der SYSTEM-Teil erlaubt insbesondere, Benutzernamen für Geräte und deren Verbindungen anzugeben (Abb.2):

```
SYSTEM ;
.
.
KANAL*6 ->MONITOR:MO100 ;
KANAL*4 ->DRUCKER: ;
KANAL*3<->PROZESSKANAL ;
          PROZESSKANAL*0->DIGITALAUS ;
                          DIGITALAUS*3->RELAIS: ;
                          DIGITALAUS*5->VENTIL: ;
          PROZESSKANAL*1<-ANALOGEIN ;
                          ANALOGEIN*2<-TEMPERATUR:;
.
.
```

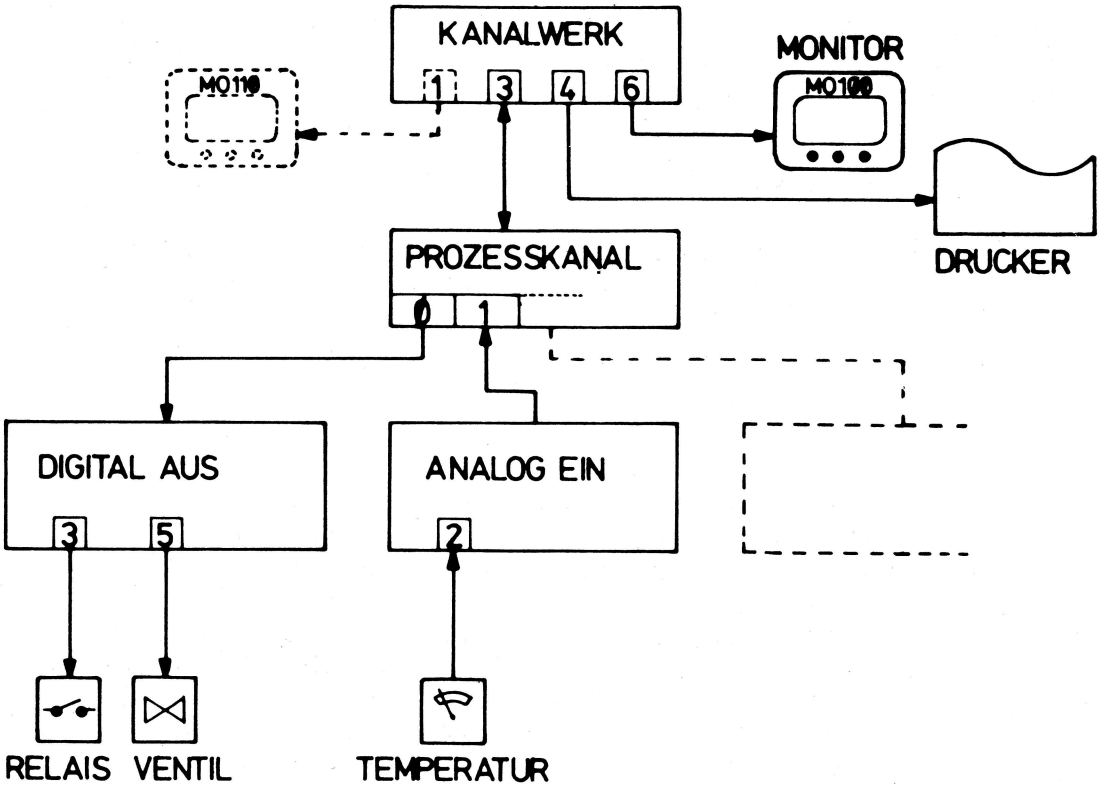


Abb. 2: Beschreibung der System-Konfiguration durch den SYSTEM-Teil

Er übernimmt teilweise die Funktion einer "Job Control Language" und entkoppelt somit die bei einer Umkonfigurierung erforderlichen Programmumstellungen vom Anwenderteil (PROBLEM-Teil). Der Ersatz des Monitors M0100 in Abb. 2 durch ein Gerät des Typs M0110 an Kanal 1 etwa würde den Austausch der entsprechenden SYSTEM-Teil-Anweisung durch

```
KANAL*1  ->  MONITOR : M0110 ;
```

erfordern.

3. PROBLEM-Teil (PROBLEM-division)

Der PROBLEM-Teil besteht aus einer Reihe von Vereinbarungen, insbesondere von Vereinbarungen derjenigen Programmteile, die gleichzeitig zum Ablauf kommen können, sog. "TASKs". TASKs können sich gegenseitig "anstoßen",d.h. Abläufe einplanen. Die erste TASK wird von außen, d.h. nicht auf Sprachebene angestoßen.

III. ALGORITHMIK

Die Sprachkonzepte der PEARL-Algorithmik basieren weitgehend auf ALGOL68 und sind soweit als möglich syntaktisch an PL/1 angeglichen worden.

1. VEREINBARUNGEN

Vom Benutzer eingeführte Programmgrößen müssen dort, wo für sie Platz anzulegen ist, deklariert werden:

```
DCL (X,Y,Z) FLOAT ;
```

Das Durchlaufen einer Deklaration kann entweder im Rahmen der Ausführung einer Task, Prozedur oder eines Kanalumsetzers geschehen, oder nicht. Im letzteren Fall stent die Deklaration "auf Modulebene" und die Größe ist für die gesamte Programmlaufzeit in dem betreffenden MODULE bekannt. Soll sie "exportiert" werden, d.h auch (in) anderen MODULEs bekannt sein, so wird die Deklaration mit dem Zusatz "GLOBAL" versehen:

```
DCL X FLOAT  GLOBAL ;
```

Sollen nur die Eigenschaften bereits existierender Größen bekannt gemacht werden, so sind diese zu spezifizieren, etwa in dem MODULE, in dem GLOBALe Größen anderer MODULEs "importiert" werden:

```
SPC X FLOAT  GLOBAL ;
```

Neben den Spezifikationen GLOBALer Größen spielen die Spezifikationen formaler Prozedurparameter eine wichtige Rolle.

2. GRUNDDATENTYPEN

Secns Grunddatentypen erlauben die Darstellung von

. Zahlen:

```
FIXED  für ganze Zahlen      : 07
FLOAT  für Gleitkommazahlen  : 7.1
```

. Ketten:

CHARACTER für Zeichenketten: 'PEARL'
BIT for Bitketten : '0101'B

. Zeiten

CLOCK für Zeitpunkte : 11 : 55 : 04
DURATION für Zeitdauern : 3 HRS 4 MIN 3 SEC

Zahlen erlauben die Angabe einer Genauigkeit, Ketten die einer Länge. Genauigkeit und Länge sind bei Operationen mit diesen Datentypen von Bedeutung.

3. Marken

PEARL kennt sowohl Markenkonstanten als auch Markenvariablen:

```
DCL MARKE LABEL ;
.
.
.
M1 : A := B + C ;
.
.
.
MARKE := M1 ;
```

Falls eine Markenvariable nur die Werte bestimmter Marken annehmen können soll, so kann der betreffende Bereich in einem Zusatz in der Deklaration angegeben werden, z.B.

```
DCL MARKE LABEL RANGE (M1, M2) ;
```

In diesem Beispiel etwa könnten der Variablen MARKE nur die Werte M1 oder M2 zugewiesen werden.

4. ZUWEISUNGSSCHUTZ

Zuweisungen an PEARL-Größen können entweder ganz oder nur bei Verwendung gewisser Bezeichner verhindert werden. Dies geschieht durch den Zusatz "INV" in einer Deklaration

```
DCL   PI   INV FLOAT  INITIAL(3.14) ;
```

oder Spezifikation

```
SPC  ZEIT  INV  CLOCK  IDENTICAL(TIME) ;
```

PI und ZEIT können also nur gelesen werden.

Das Sprachmittel Zuweisungsschutz läßt sich mit Vorteil zur Programmstrukturierung anwenden, etwa, wenn Größen nur in einem Modul verändert, in anderen jedoch gelesen werden können:

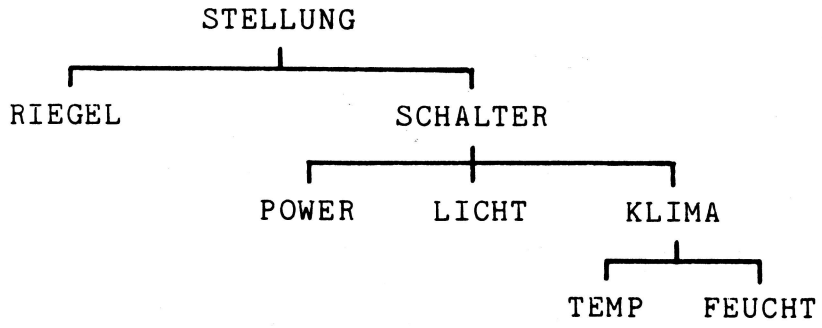
```
MODULE ;
PROBLEM ;
.
.
.
DCL SCHALTER FIXED GLOBAL ;
.
.
.
MODEND ;

MODULE ;
PROBLEM ;
.
.
.
SPC SCHALTER  INV FIXED GLOBAL ;
.
.
.
MODEND ;
```

5. ZUSAMMENGESETZTE GRÖSSEN

Aus den Grunddaten und den meisten anderen Programmgrößen lassen sich Felder und Verbunde zusammensetzen:

Insbesondere können Verbunde aus Verbunden aufgebaut und Teilverbunde getrennt angesprochen werden, wodurch sich Datenhierarchien grosser Komplexität übersichtlich handhaben lassen.

```
DCL STELLUNG STRUCT
      [RIEGEL SEMA, SCHALTER STRUCT
                      [POWER BIT(1),
                       LICHT BIT(1),
                       KLIMA STRUCT
                         [TEMP FLOAT,
                          FEUCHT FIXED]]] ;
```



6. ZEIGER

Zeiger auf PEARL-Größen sind typspezifisch:

```
DCL  MATZEIG  REF  ( , )  FLOAT ;
```

Damit kann MATZEIG auf zweidimensionale Matrizen von FLOAT-Größen zeigen.

Nicht auf alle PEARL-Größen kann mit einem Zeiger verwiesen werden. Zum Beispiel kennt PEARL keine Zeiger auf Zeiger oder Prozeduren.

7. PROZEDUREN

Die Deklaration von PEARL-Prozeduren erlaubt eine präzise Steuerung des Parameter- und Resultat-Übergabemechnismus. Für jeden Parameter läßt sich angeben, ob die Prozedur von seinem Wert an der Aufrufstelle oder seiner "Identität" (d.h. seiner Adresse) Gebrauch macht.

Im ersteren Fall (INITIAL-Mechanismus) wird eine Kopie der aktuellen Parameter erstellt, im letzteren (IDENTICAL-Mechanismus) wird nur ein neuer Bezeichner im Gültigkeitsbereich des Prozedurblocks vereinbart. Kombiniert mit selektivem Parameter-Zuweisungsschutz innerhalb des Prozedurblocks lassen sich vielfältige Zugriffe auf Parameter erreichen.

In

```
P : PROC  (FELD  ( , ) FLOAT IDENTICAL ,
           ZEIT INV CLOCK IDENTICAL ,
           I    FIXED
          ) ;
    .
    .
    .
    END ;
```

ist FELD eine zweidimensionale FLOAT-Matrix und ZEIT eine CLOCK-Größe, die beide beim Prozeduraufruf nicht kopiert werden (IDENTICAL). "ZEIT" ist überdies zuweisungsgeschützt, kann also innerhalb der Prozedur nicht verändert werden (aber z.B. in einer anderen Task). Die FIXED-Größe "I" wird kopiert (die Angabe INITIAL ist default),d.h. die Prozedur macht von ihrem Wert an der Aufrufstelle Gebrauch.

Funktionsprozeduren, d.h. Prozeduren, die einen Wert zurückliefern, sind durch das Attribut

"RETURNS"

,gefolgt vom Typ der zurückgelieferten Größe, gekennzeichnet:

```
SINUS : PROC ( X FLOAT )  RETURNS (FLOAT) ;
        .
        .
        .
        END ;
```

Weitere Attribute machen Aussagen über Mehrfachbenutzbarkeit (REENT) und Direkteinbettung (INLINE) des Prozedurcodes.

8. VEREINBARUNG NEUER DATENTYPEN

PEARL ermöglicht dem Programmierer die Vereinbarung von Bezeichnern für neue Datentypen wie sie etwa durch Verbunde definiert werden:

```
TYPE COMPLEX STRUCT [RE FLOAT , IM FLOAT] ;
.
.
.
DCL   C  COMPLEX ;
```

Ferner kann ein Typ-Bezeichner für einen von einer Reihe von Typen stehen:

```
TYPE  TRANSFER ONEOF(COMPLEX, CHARACTER(16)) ;
```

etwa als Abkürzung für in einen Übertragungskanal einspeisbare Datentypen, der im obigen Beispiel nur den Strukturtyp COMPLEX und Zeichenketten aufnehmen kann.

9. DEFINITION NEUER OPERATIONEN

Die Konstruktion eigener Datentypen wird durch die Möglichkeit, auf sie abgestimmte Operatoren zu definieren, ergänzt.

```
OPERATOR + (A INV COMPLEX, B INV COMPLEX)
             RETURNS (COMPLEX) ;

RETURN ( [A.RE + B.RE , A.IM + B.IM] ) ;

END ;
```

Mittels einer Operator-Deklaration lassen sich Bezeichner oder Symbole für beliebige monadische oder dyadische Operatoren einführen, die in ihrer Semantik "generic" Funktionen ähneln, d.h. Operatoren mit gleicher Bezeichnung werden über die Argumenttypen unterschieden.

IV. EIN- UND AUSGABE

Der zu erwartenden Vielfalt an Prozeßperipherie wurde in PEARL durch ein besonders anpassungsfähiges E/A-System entsprochen. Es besteht aus einem Netzwerk von Datenwegen mit den Komponenten

. DATENSTATIONEN, als Verallgemeinerung von reellen oder virtuellen Peripheriegeräten bzw. E/A-Kanälen

. KANALUMSETZER ("Interfaces"), zur Abbildung von Datenstationen verschiedener Eigenschaften aufeinander mit der Möglichkeit, benutzereigene Formate zu definieren.

Abbildung 3 zeigt schematisch die Umsetzung eines benutzer-definierten Satzes von E/A-Kanälen auf durch das System vorgegebene E/A-Kanäle vermittels eines auf Sprachebene formulierten Kanalumsetzers.

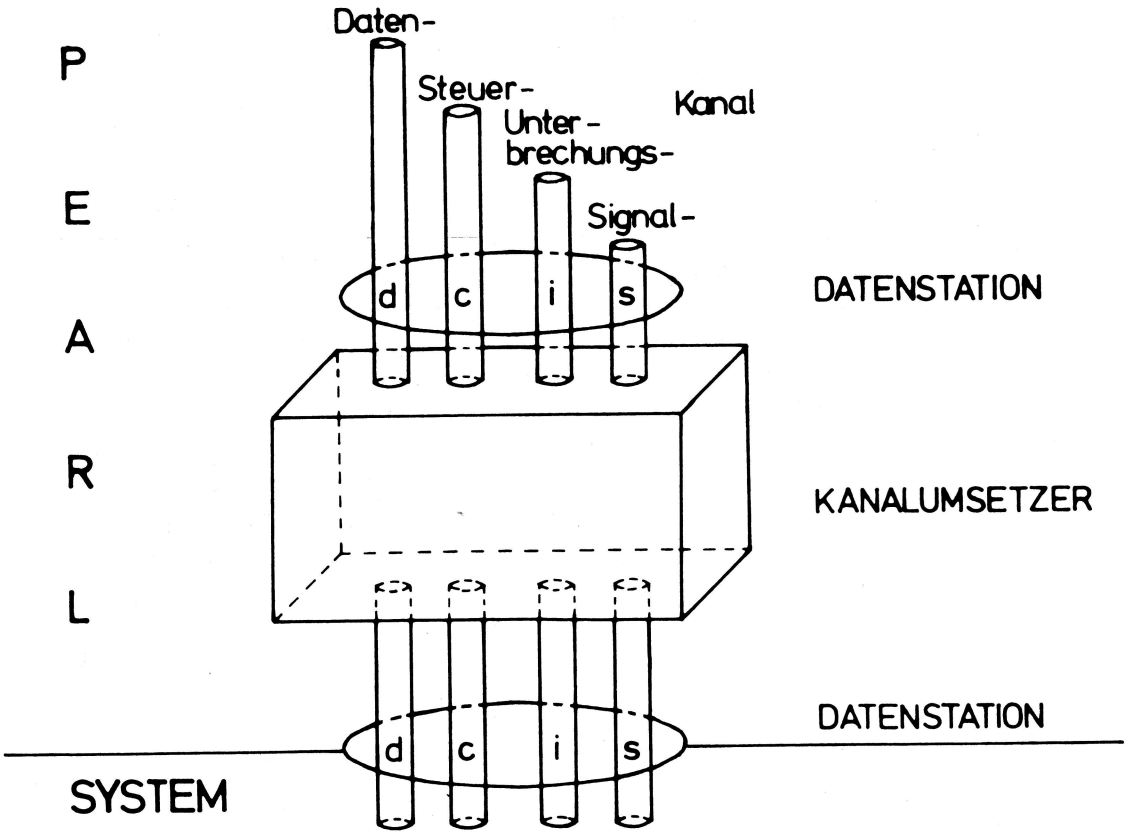


Abb. 3: Datenstationen und Kanalumsetzer

1. DATENSTATIONEN

Ein E/A-Gerät sowohl der Standard- als auch der Prozeßperipherie wird in PEARL als ein Satz von maximal vier Kanälen betrachtet:

- ein Daten-Kanal nimmt die Werte von "übertragbaren" PEARL-Größen auf oder liefert solche Werte ab. Naturgemäß sind nicht alle PEARL-Größen in Datenkanälen übertragbar, im wesentlichen sind es die Grunddatentypen und aus ihnen zusammengesetzte Größen. Weiterhin blokkiert ein Zuweisungsschutz (INV-Attribut) die Eingabe auf die betreffenden Größen.

- ein Steuer-Kanal nimmt zur Steuerung des Gerätes nötige Informationen auf. Ihrer besonderen Bedeutung entsprechend, werden sie in PEARL über einen eigenen Datentyp gehandhabt. Die Ansprache des Steuer-Kanals erfolgt durch Größen des Typs "CONTROL".

- ein Unterbrechungs-Kanal meldet Ereignisse vom Typ "INTERRUPT" (siehe Abschnitt V.)

- ein Signal-Kanal meldet Ereignisse vom Typ "SIGNAL" (siehe Abschnitt V.)

Um den Umgang mit einem E/A-Gerät auf Sprachebene überwachen zu können, müssen die Eigenschaften seiner Kanäle spezifiziert werden. Dies geschieht in der Spezifikation einer Datenstation:

```
SPC DRUCKER DATION OUT AI PHIC CONTROL(ALL) ;
```

Sie enthält z.B. Angaben über

- Richtung des Datenflusses in Bezug auf Zentraleinheit bzw. Hauptspeicher (IN,OUT,INOUT)

- Möglichkeiten des direkten Verkehrs mit anderen Peripheriegeräten (SOURCE, SINK)

- Darstellungsart der übertragenen Daten (Interndarstellung oder Wandlung in externe Grössen, z.B. abdruckbare Zeichen (ALPHIC), logische Pegel (BASIC), graphische Symbole (GRAPHIC))
- Übertragungseinheit (z.B. eine Zeile oder ein Bildschirminhalt)

- Strukturierung der übertragenen Daten (z.B. Dateigliederung)

. Zugriffsmethode (FORWARD, FORBACK, DIRECT)

. Fortschaltungsverfahren (STREAM, CYCLIC)

Diese Attribute einer Datenstation ermöglichen umfassende Prüfungen von E/A-Anweisungen i.a. bereits zum Zeitpunkt der Übersetzung.

2. KANALUMSETZER

Aufbauend auf den von einer Implementierung zur Verfügung gestellten Geräten (genauer: Treiberschnittstellen) lassen sich virtuelle Geräte definieren, die adäquatere (z.B. problemnähere) Eigenschaften besitzen. Die erforderlichen Umsetzungen lassen sich in einer Routine beschreiben, die prozedurartigen Charakter besitzt.Diese zwischen zwei Datenkanäle schaltbare Umsetzungsroutine (Abb.3) ermöglicht beispielsweise die

. Generierung von bzw. Reaktion auf INTERRUPTs und SIGNALe

. Selektion von Unterkanälen

. Dateieröffnungs- und -abschlußbehandlung

. Anpassung von Übertragungseinheiten (Pufferung)

. Einfügen von Kontrollinformation in bzw. seine Trennung aus dem Datenstrom.

3. E/A-ANWEISUNGEN

Der eigentlichen Benutzung der E/A-Kanäle einer Datenstation geht eine Operation voraus, die einen Datenweg aufbaut und eine weitere, die seine Benutzung durch verschiedene Tasks synchronisiert. Es gibt daher drei Gruppen von E/A-Anweisungen:

. Auf- und Abbau von Datenwegen

Die CREATE-Anweisung verknüpft Datenstationen über Kanalumsetzer, z.B. die durch Abbildung 3 gegebene Anordnung:

```
CREATE OUTFILE UPON DRUCKER USING PRINTERFACE ;
```

Entsprechend löst die DELETE-Anweisung solche Verknüpfungen wieder auf:

```
DELETE OUTFILE ;
```

. Synchronisation der Benutzung von Datenwegen

Ein durch CREATE-Anweisungen geschaffener Datenweg kann von einer Task exklusiv belegt oder von mehreren Tasks benutzt werden. Entsprechende Angaben sind in der OPEN-Anweisung zu machen, z.B.

```
OPEN OUTFILE BY EXCLUSIVE ;
```

Der Verzicht auf die Benutzung eines Datenweges wird durch die CLOSE-Anweisung ausgedrückt.

. Die eigentliche Benutzung des Datenweges geschieht durch Übertragungs-Anweisungen

Abhängig von den angesprochenen Datenstationen gibt es verschiedene Arten von Übertragungsanweisungen. Ein graphisches Ausgabegerät wird z.B. mit "DRAW" angesprochen, der in 1. spezifizierte DRUCKER etwa durch

```
PUT 'ABC' TO DRUCKER BY SKIP, A(3) ;
```

Die beiden Steuergrößen "SKIP" und "A(3)" sprechen den Steuer-Kanal an, während die Zeichenkette 'ABC' in den Daten-Kanal der Datenstation "DRUCKER" eingespeist wird.

V. ECHTZEITELEMENTE

Herausragendes Merkmal einer Echtzeit-Programmiersprache im Vergleich zu Sprachen für kommerzielle oder technisch-wissenschaftliche Anwendungen ist die Möglichkeit zur Formulierung und Abwicklung unabhängiger (asynchroner) Programmabläufe.

PEARL stellt eine Reihe von Sprachmitteln zu Verfügung, die die Handhabung von Nebenläufigkeiten sowie die Reaktion auf Unterbrechungen, Fehlermeldungen u.s.w. erleichtern.

1. TASKS

Ähnlich wie mit einer Prozedur ist mit einer Programmgröße vom Typ TASK ein Stück Code verknüpft :

```
FILLING : TASK ;
            .
            .
            .
          END ;
```

Dem Prozedur-Aufruf entspricht die Aktivierung der Task, wobei die Ausführungen verschiedener Tasks untereinander zeitlich unkorreliert sind:

```
ACTIVATE FILLING ;
```

Ihre Ausführungsgeschwindigkeit kann qualitativ durch Angabe einer Priorität beeinflußt werden.

```
REPORT : TASK PRIO 25 ;
           .
           .
           .
         END ;
```

Die Ausführung von Tasks kann

. für unbestimmte Zeit unterbrochen werden:

```
SUSPEND  FILLING ;
```

. nach einer Unterbrechung fortgesetzt werden:

```
CONTINUE  FILLING ;
```

. für eine bestimmte Zeispanne unterbrochen werden:

```
AFTER 10 MIN RESUME REPORT ;
```

. ganz abgebrochen werden:

```
TERMINATE  FILLING ;
```

. Ferner lassen sich alle Einplanungen für eine Task streichen:

```
PREVENT REPORT ;
```

Aktivierungen und andere Task-Operationen können über eine Einplanungsklausel ("schedule")

. an Zeitpunkte, Zeitdauern oder beides

```
AT 9:0:0  ALL  2 HRS  UNTIL  18:0:0
ACTIVATE REPORT ;
```

. an das Auftreten von Ereignissen

```
WHEN  EMPTY  CONTINUE  FILLING ;
```

geknüpft werden.

2. SYNCHRONISATION

Falls Operationen verschiedener Tasks zeitlich korreliert werden müssen, z.B. weil sie auf gemeinsame Daten oder andere Betriebsmittel zugreifen, so lassen sich Synchronisationsalgorithmen mit Hilfe von SEMAphor- oder BOLT-Größen erstellen.

. Die Semaphor-Operationen REQUEST, RELEASE eignen sich besonders zur Sicherstellung der alleinigen Benutzung von Betriebsmitteln:

```
DCL  CLEARANCE  SEMA ;
.
.
.
REQUEST  CLEARANCE ;
```

führt zur Suspendierung sofern eine andere Task das Semaphor erfolgreich angefordert hat und es noch nicht durch

```
RELEASE  CLEARANCE ;
```

wieder freigegeben wurde.

Mit Hilfe von BOLT-Operationen läßt sich z.B. das Wechselspiel von Sperrung und Freigabe bei gemeinsamem (lesendem) und alleinigem (schreibendem) Zugriff formulieren:

```
DCL  ZUGRIFF  BOLT ;
.
.
.
RESERVE  ZUGRIFF ;
```

verlangt exklusiven Zugriff zu dem durch die BOLT-Größe geschützten Betriebsmittel. Es führt zur Suspendierung, sofern andere Tasks gerade das Betriebsmittel benutzen, verhindert jedoch erneute Zugriffe von Tasks mit niedrigerer Priorität, etwa lesenderweise durch

```
ENTER  ZUGRIFF ;
```

Sobald alle "Leser" auf das Betriebsmittel verzichten, was durch die Operation

```
LEAVE  ZUGRIFF ;
```

geschieht, kann die RESERV(E)ierung wirksam werden.

Das Ende der Exklusiv-Belegung wird durch

```
FREE  ZUGRIFF ;
```

angezeigt, wonach (prioritätsgerecht) neue ENTER- oder RESERVE-Operationen möglich werden.

3. EREIGNISSE

PEARL erlaubt, zwei Typen von Ereignissen zu definieren und darauf zu reagieren:

. Unterbrechungen (INTERRUPTs) sind ansynchron zum Programmablauf auftretende Ereignisse, die i.a. im SYSTEM-Teil als von einem (virtuellen) Gerät herrührend vereinbart werden und auf die durch TASK-Operationen, z.B. Aktivierung, reagiert werden kann:

```
SPC  EMPTY  INTERRUPT ;
.
.
.
WHEN  EMPTY  ACTIVATE  FILLING ;
```

. SIGNALe werden wie INTERRUPTs i.a. im SYSTEM-Teil vereinbart; es sind Ereignisse, die in unmittelbarem Zusammenhang mit einer Operation stehen (typisch Fehlermeldungen wie Overflow oder End-of-File) und auch nur derjenigen Task zugeleitet werden, die diese Operation ausgeführt hat. Diese kann auf ein SIGNAL durch Ausführung eines ON-Blocks reagieren:

```
SPC  EOF  SIGNAL ;
.
.
.
ON  EOF : BEGIN ;
          .
          .  /* REAKTION AUF END-OF-FILE */
          .
          END ;
```

Unterbrechungen und Signale lassen sich auf Sprachebene auslösen:

```
TRIGGER  EMPTY ;

INDUCE  EOF ;
```

was für Test- und Simulationszwecke von besonderer Bedeutung ist.

L I T E R A T U R

/1/ Full PEARL Language Description
Gesellschaft für Kernforschung mbH, Karlsruhe,
PDV-Bericht KFK-PDV 130, 1977

/2/ Basic PEARL Language Description
Gesellschaft für Kernforschung mbH, Karlsruhe,
PDV-Bericht KFK-PDV 120, 1977
(die deutsche Übersetzung davon ist als
PDV-Bericht KFK-PDV 121 erschienen)

/3/ MARTIN, T.: Die Förderung von PEARL im Projekt PDV
(Prozeßlenkung mit Datenverarbeitungsanlagen)
des 2. und 3. DV-Programms des BMFT.
In: Tagungsband zum Aussprachetag PEARL
in Augsburg, März 1977, 2. Auflage,
Gesellschaft für Kernforschung mbH,
Karlsruhe, PDV-Bericht KFK-PDV 110, 1977