Portables PEARL-Programmiersystem von Werum, Charakterisierung

H. Windauer

1. Implementierter Sprachumfang

Der implementierte Sprachumfang umfaßt Basic PEARL nach DIN 66253 (Entwurf, Teil 1) sowie die folgenden über Basic PEARL hinausgehenden Sprachelemnete. Alle diese Sprachelemente sind in Full PEARL enthalten.

Datentypen

- Referenzen (REF)
- Neudefinierte Typen (TYPE)
- BOLT-Variablen
- Bereiche (Felder, Arrays) von SEMA-, BOLT-,REF-, DATION-Objekten, von Strukturen (STRUCT) und von Variablen neu definierten Typs
- Mehr als drei Bereichsdimensionen
- Untere Dimensionsgrenzen von Bereichen beliebig, auch negativ
- Bereiche, Strukturen, REF - Variablen und Variablen neu definierten Typs als Strukturkomponenten
- B2 bei Bitkettenkonstanten.

Vereinbarungen

- Moduln mit Bezeichner (z.B.MODULE (Zentral))
- Global-Attribut mit Modulbezeichner (z.B...GLOBAL (Zentral))
- Vereinbarungen neuer Typen (TYPE)
- Vereinbarungen neuer Operatoren (OPERATOR)
- Rang-Vereinbarung (PRECEDENCE)
- Reihenfolge der Vereinbarungen beliebig
- Vereinbarung von Prozeduren nicht nur auf Modulebene, sondern auch in Tasks, Prozeduren, Blöcken und Wiederholungen
- Als Prozedurparameter auch Objekte vom Typ SEMA, BOLT, IRPT, SIGNAL und REF sowie Objekte neu vereinbarten Typs
- Als Ergebnisse von Funktionsprozeduren auch Objekte vom Typ REF, STRUCT und neu vereinbarten Typs
- Für INIT und IDENT sind auch die Langformen INITIAL und IDENTICAL zulässig.

Anweisungen

- Auf der linken Seite von Zuweisungen auch dereferenzierte Referenz-Variablen und Zeichenkettenausschnitte
  (z.B. K.CHAR (J):='N';)
- Beim ACTIVATE kann die Startbedinung mit einer Frequenz und/oder AFTER Ausdruck§Dauer kombiniert sein
  (z.B. WHEN Interrupt ATER Dauer ALL Dauer DURING Dauer ACTIVATE Task;)
- SUSPEND kann auch auf fremde Tasks wirken
- Prioritätsangabe beim CONTINUE
  (zum Ändern der eigenen Priorität, z.B.CONTINUE PRIO 2;)
- Listen von SEMA - Varibablen hinter REQUEST und RELEASE
- BOLT - Anweisungen ENTER, LEAVE, RESERVE, FREE
- Listen von BOLT - Variablen in BOLT - Anweisungen
- TRIGGER - Anweisung

Ausdrücke

- Zeichenkettenausschnitte, variable Kettenausschnitte (z.B. XKOORD:=EINGABE.BIT (I:I+3); AUSGABE.CHAR (J):=STRING.CHAR (K);)
- Dereferenzierung (CONT)
- Bedingter Ausdruck auf Wertbasis
  (z.B. A:=IF B>1 THEN B ELSE C FIN;)
- Operatoren LWB und UPB auch monadisch.

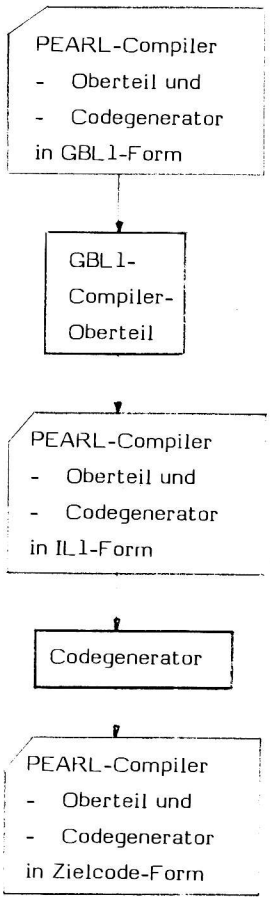
Ein/Ausgabe

- Als Typ der Übertragungsdaten auch Strukturen, neu vereinbarte Typen und ALL
- Bereiche von benutzer-definierten Datenstationen
- Als Open-Parameter auch CAN und PRM
- Close-Parameter CAN, PRM.

Systemteil

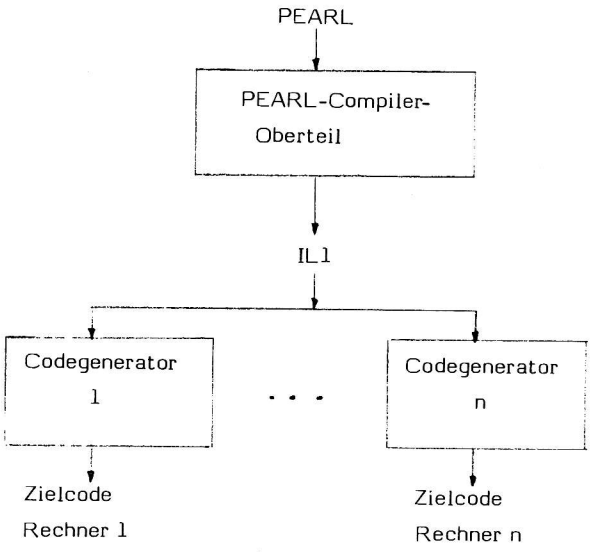
- Beliebige Reihenfolge der Verbindungen
- Spiegelbildliche Umkehrung einer Notation zugelassen
- Bei Anschlüssen hinter * auch Bezeichner§Anschlußstelle und /Schrittweite zugelassen.

Bezüglich Compiler ist dieser Sprachumfang allen Rechnern implementiert, bei denen der PEARL-Compiler von WERUM eingesetzt wird. Bei einzelnen Rechnern können sich jedoch Einschränkungen durch die Größe des Laufzeitsystems ergeben. So müssen auf Siemens 404/3 (maximaler Ausbau 64 KB) Einschränkungen z.B. bezüglich Signalverarbeitung und File-Handling beachtet werden.



2. Kurzdarstellung der verwendeten Compiler-Technologie

Der portable PEARL-Compiler von WERUM besteht aus einem analytischen Teil ("Oberteil"), der ein PEARL-Programm in die rechnerunabhängige Zwischensprache IL1 (Intermediate Language 1) übersetzt und einem codegenerierenden Teil ("Codegenerator"), der das PEARL-Programm aus der IL1-Form in den Zielcode überführt. Der Oberteil ist rechnerunabhängig und deshalb nur einmal programmiert; die rechnerabhängigen Codegenaratoren werden für jeden Typ Rechner, auf dem PEARL-Programme ablaufen sollen, neu erstellt.



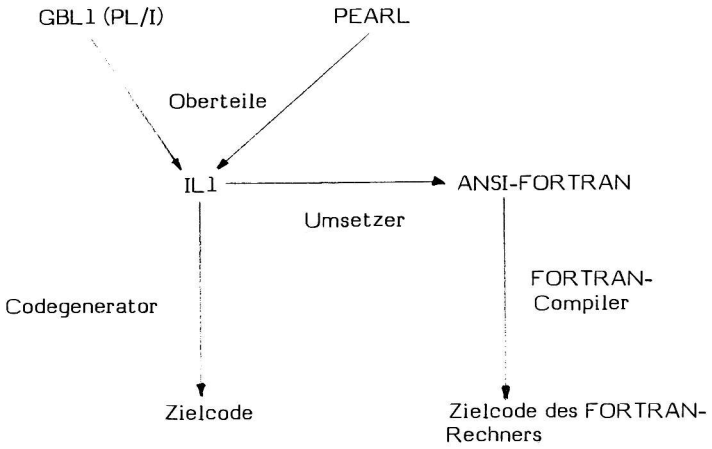
Der Compiler-Oberteil und die Codegeneratoren sind in GBL1, einem PL/I-Subset, programmiert. Für diesen PL/I-Subset steht ebenfalls ein Compiler-Oberteil GBL1→ IL1 zur Verfügung. Mit seiner Hilfe und dem entsprechenden Codegenerator werden der PEARL-Compiler-Oberteil und der Codegenerator selbst in den Zielcode (Assembler oder BRF) des Zielrechners übersezt.
Diese Übersetzung wird in der Regel auf einem Entwicklungsrechner von WERUM durchgeführt, der über einen GBL1-Compiler verfügt (z.Zt. Siemens 330).

Der PEARL-Compiler kann also auf dem Zielrechner installiert werden, ohne daß dieser über einen PL/1-Compiler verfügt.

Zum Ablauf der übersetzten PEARL-Programme auf dem Zielrechner ist natürlich zudem die Implementierung von PEARL-spezifischen Laufzeit- und Betriebssystemfunktionen erforderlich.

Der PEARL-Compiler eignet sich aber auch für Cross-Compilierung: Aufgrund der skizzierten Compiler-Technologie kann er auf allen Rechnern installiert werden, die über einen PLI/-Compiler (mit einem GBL1 umfassenden Sprachumfang; Beispiel: IBM) oder einen Codegenerator verfügen.

Über einen Umsetzer von IL1 nach ANSI-FORTRAN kann der PEARL-Compiler außerdem auf FORTRAN-Rechner übertragen werden, um dort als Cross-Compiler (Beispiel:Siemens 7.760) für andere Zielrechner oder als Compiler für diesen FORTRAN-Rechner (Beispiel: HP 3000) zu arbeiten.



3. Existierende Komponenten des PEARL-Programmiersystems

Das portable PEARL-Programmiersystem von WERUM besitzt folgende Komponenten:

- Compiler (Oberteil und Codegenerator)
- Betriebssystem-Kern
- Laufzeitpaket für binäre E/A
- Laufzeitpaket für formatierte E/A
- Testsystem
- Datenhaltungssystem

Bei den bisherigen Implementierungen wurden die Laufzeitroutinen für die PEARL-spezifische Arithmetik und Behandlung von Bit- und Zeichenketten, Prozeduren und Feldern etc. jeweils rechnerabhängig erstellt.
Im übrigen werden die Standardkomponenten des Zielrechners (z.B. Binder, Editor) eingesetzt.

3.1 Compiler

Der PEARL-Compiler übersetzt PEARL-Programme je nach Rechner in Assembler oder verschiebbaren Bindercode. Die Handhabung der vielfältigen Compiler-Parameter ist oder wird der Bedienungsart angepaßt, die auf dem jeweiligen Zielrechner üblich ist.

Mittels solcher Compiler-Parameter können u.a. Listings der Quelle und des Übersetzungsergebnisses verlangt werden, wobei im Assembler- oder Bindercode-Listing Hinweise auf die Ursprungszeilen im Quellprogramm eingefügt sind. Außerdem erstellt der Compiler auf Wunsch eine Cross-Referenz-Liste aller benutzten Objekte mit Angaben der Quellzeilen-Nummern für ihre Definition und Benutzung.
Durch Compiler-Parameter kann die Überwachung von Feldgrenzen und Referenzen zur Laufzeit ein- oder ausgeschaltet werden.

Der Compiler analysiert Programme umfangreich und recht genau. Die Fehlermeldungen erfolgen im Klartext mit Angaben der Quellzeilen-Nummer (423 verschiedene Fehlermeldungen sind möglich).

Ein Preprozessor gestattet das Einfügen von Programmtexten aus Files (%INCLUDE) und bedingte Compilierung (%IF).

Die Systemteil-Auswertung wird durch eine sogenannte Konfigurationsliste gesteuert, die alle Konfigurationsmöglichkeiten des Zielrechners beschreibt. Sollen diese Möglichkeiten erweitert werden, z.B. beim Anschluß eines bisher nicht vorgesehenen Gerätes, so kann der Anwender selbst mit einfachen Mitteln diese Konfigurationsliste anpassen und dem Compiler (sogar dynamisch) bekannt machen.

Auf Wunsch ist der Compiler in einer Version erhältlich, die bei entsprechender Einstellung eines Compiler-Parameters folgende Optimierungen durchführt:

- Nur einmalige Berechnung der Adressen von indentischen Strukturkomponenten innerhalb einer PEARL-Anweisung.
- Nur einmalige Berechnung gemeinsamer Teilausdrücke.
- Direktes Absetzen fest indizierter Feldansprachen, d.h. keine Index-Berechnung zur Laufzeit in diesen Fällen.
- Rechnerabhängige Optimierungen.

Der Compiler ist in GBL1, einem echten PL/I-Subset, programmiert.

3.2 Betriebssystem-Kern

Für die PEARL-gerechte Organisation von Task und deren Ablauf,für die Synchronisation und die Prozeß-E/A steht ein in GBL1 programmierter, rechnerunabhängiger Betriebssystem-Kern BAPAS-K zur Verfügung, der mittels des GBL1-Compilers automatisch auf den Zielrechner transportiert werden kann.

Dort werden die offenen Schnittstellen "per Hand" geschlossen, wobei BAPAS-K auf dem Gast-Betriebssystem aufgesetzt oder ohne Gast-Betriebssystem implementiert werden kann.

3.3 Laufzeitpaket für binäre E/A

Das Laufzeitpaket BAPAS-FILE enthält alle Laufzeitroutinen, die für einen PEARL-gerechten Ablauf der PEARL-Anweisungen für binäre E/A einschließlich File-Organisation erforderlich sind. Die Schnittstelle dieser rechnerunabhängigen Routinen zu den Zielrechnern wird durch geeignete Kontrollsätze und Treiber-Aufrufe gebildet.

Das Paket ist in PEARL programmiert und kann somit automatisch mittels des PEARL-Compilers auf die Zielrechner transportiert werden,

3.4 Laufzeitpaket für formatierte E/A

Analog zu BAPAS-FILE enthält BAPAS-FORMEA alle Laufzeitroutinen, die für einen PEARL-gerechten Ablauf der PEARL-Anweisungen für formatierte E/A erforderlich sind, in rechnerunabhängiger Form.

Das Paket ist sehr modular in PEARL programmiert und kann mittels des PEARL-Compilers automatisch auf die Zielrechner transportiert werden.

3.5 Testsystem

Für den Test von PEARL-Programmen auf Ziel- und Cross-Rechnern ist ein Testsystem mit verschiedenen Ausbaustufen erhältlich; seine Bedienkommandos sind an PEARL angelehnt. Es ist portabel in GBL1 und PEARL programmiert.

Stufe 1

Die erste Ausbaustufe umfaßt die folgenden Testhilfen auf PEARL-Sprachebene:

- Zeilen-Trace
- Breakpoints auf Zeilen
- Ausgabe von Werte von Variablen

Diese Ausbaustufe kann auf dem Zielrechner installiert werden.

Stufe 2

Die zweite Ausbaustufe bietet auf PEARL-Sprachebene die Testhilfen

- Zeilen-, Marken- und Call-Trace
- Breakpoints auf
  - Zeilen und Marken
  - Eingänge und Ausgänge von Prozeduren
- Ausgabe und Veränderung der Werte von Variablen
- Ausgabe und Veränderung der Zustände von Tasks, Semas und Bolts
- Automatische Erkennung von
  - falschen Indizierungen bei Zugriffen auf Feldelemente oder Kettenausschnitte,
  - E/A-Fehlern,
  - rekursiven Aufrufen von Prozeduren,
  - Aktivierungen aktiver Tasks und
  - Dereferenzierungen der Null-Referenz.

Diese Ausbaustufe läßt sich auf größeren Zielrechnern und auf Cross-Rechnern installieren. Bei der Cross-Version werden das PEARL-Betriebssystem des Zielrechners (unter Verwendung von BAPAS-K) und die Zeitachse auf dem Cross-Rechner insoweit simuliert, daß das Zusammenspiel der Tasks in zeitlich richtiger Reihenfolge erfolgen kann.

Stufe 3

Die dritte Ausbaustufe wird in der Regel auf einem Cross-Rechner installiert. Sie enthält zusätzlich zur zweiten Ausbaustufe die folgenden Testhilfen:

- Operationsgenaue Simulation des Zeitverhaltens von Zielrechnern
- Simulation der E/A von Zielrechnern mittels
  - E/A im Dialog mit dem Bediener
  - Gegentasks
  - Files mit Testdaten
- Breakpoints auch auf
  - zeitliche Ereignisse (analog PEARL-Schedules)
  - Interrupts
  - E/A-Anweisungen
- Deadlock-Analyse
- Interrupt-Anweisungen

3.6 Datenhaltungssysteme

Zur Unterstützung der PEARL-Programmierung von Automationssystemen mit großem Datenhaltungsanteil entwickelt WERUM ein Echtzeit-Datenhaltungssystem, das simultan durch Bediener im Dialog und durch PEARL-Tasks angesprochen werden kann.

Das Datenhaltungssystem ist offen, d.h. bei Beibehaltung seiner Struktur und Schnittstellen können seine Zugriffsverfahren ausgetauscht werden.

Es wird portabel in PEARL programmiert.

Die erste Ausbaustufe mit einem verfeinerten B-Baum-Verfahren ist realisiert.
(Die Entwicklung des Systems erfolgt mit Mitteln des BMFT im Rahmen eines Forschungsvorhabens des Projekts Fertigungstechnik (PFT) der Kernforschungszentrum Karlsruhe GmbH.)

4. Übersetzungsrechner

Der PEARL-Compiler ist auf Rechnern folgenden Typs installiert (die Zahlen in Klammern geben den Ausbau der Rechner an):

- Amdahl 470/6
- Hewlett-Packard HP 3000 (512 KB)
- Hewlett-Packard HP 1000 (64 KB)
- Norsk Data NORD 10 S (128 KB)
- Siemens 310 (64 KB)

- Siemens 330 (128 KB)
- Siemens 7.760
- Siemens 404/3 (64 KB)

Er kann relativ einfach und schnell (d.h. ohne Erstellung eines Codegenerators) auf PL/I- und FORTRAN-Rechnern installiert werden.

5. Zielrechner

Die vom PEARL-Compiler übersetzten Programme sind je nach Codegenerator (oder Umsetzer) ablauffähig auf folgenden Rechnern (die Zahlen in Klammern geben den Ausbau an):

- Hewlett-Packard HP 3000 (512 KB)
- Hewlett-Packard HP 1000 (64 KB);
  in Zusammenarbeit mit der TU Berlin;
  Termin: Ende 1980
- Norsk Data NORD 10 S /NORD 100 (128 KB);
  in Zusammenarbeit mit Norsk Data;
  Termin Herbst 1980
- RDS (Really Distributed System: Mehrrechnersystem mit räumlich verteilten (Mikro-) Prozessorstationen mit Siemens 310-Befehlsatz);
  in Zusammenarbeit mit den Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB), Karlsruhe
- Siemens 330 (128 KB)
- Siemens 404/3 (64 KB);
  in Zusammenarbeit mit mbp.

In allen Fällen werden die Standard-Betriebssysteme der Hersteller sowie BAPAS-FORMEA benutzt; BAPAS-K wird bei HP 3000, Siemens 330 und Siemens 404/3, BAPAS-FILE bei HP 3000 und Siemens 330 eingesetzt.

6. Lieferform, Schulungsmaterial, Benutzerhandbücher, Wartungsleistungen

Lieferform

Das Programmiersystem wird je nach Bestellung ganz oder komponentenweise auf Magnetplatte, Magnetband oder Floppy Disk in der Form geliefert, die der Codegenerator, FORTRAN-Umsetzer oder PL/I-Comiler erzeugt. Auf Wunsch können auch Quellen und technische Dokumentation geliefert werden.

Schulungsmaterial

Der implementierte Sprachumfang ist in dem Buch
PEARL.Beschreibung mit Anwendungsbeispielen.
Vieweg 1978
von Wulf Werum und Hans Windauer
beschrieben.

Auf Wunsch werden mehrtägige Schulungskurse beim Kunden durchgeführt.

Benutzerhandbücher

Für die Handhabung des Programmiersystems können allgemeine und/oder rechnerspezifische Benutzerhandbücher geliefert werden.

Wartungsleistungen

Die Garantiezeit beträgt ein Jahr nach Abnahme. Anschließend wird ein Wartungsvertrag angeboten, der die schnelle Beseitigung dringender Störungen und die Lieferung von Releases umfaßt.

7. Referenzen über Einsätze

Amdahl 470/6:	Gesellschaft für Reaktorsicherheit (GRS) mbH, Garching:Sicherheitsrelevante Analysen
HP 1000:	Prozeßrechnerverbund der TU Berlin, Berlin: Steuerung von Experimenten und Prozessen
HP 3000:	Institut für Rundfunktechnik,München: Entwicklung rundfunkspezifischer Software
NORD 100:	Institutt for Energiteknikk,Halden, Norwegen: Entwicklung von portabler Software für Analysen,Simulationen, Steuerungen TU Braunschweig: Steuerung eines Datenbankrechners
NORD 10 S:	WERUM,Lüneburg:Entwicklung von Basis-Software
RDC-System:	IITB,Karlsruhe: Entwicklung von Anwendungssoftware Thyssen AG,Duisburg:Regelung von Tieföfen
Siemens 310:	IITB,Karlsruhe: Entwicklung von Anwendungssoftware
Siemens 330:	IITB,Karlsruhe:Cross-Compilierung für RDC WERUM,Lüneburg: Entwicklung von Basis-Software
Siemens R30:	IITB,Karlsruhe:Entwicklung von Anwendungssoftware
Siemens 404/3:	DFVLR,Oberpfaffenhofen: Entwicklung von Software für SPACELAB
Siemens 7.760:	IITB,Karlsruhe: Cross-Compilierung für RDC.