Das PEARL-Übersetzungssystem von Dornier-System, Friedrichshafen

Dr. M. Ammann, Dr. P. Elzer

1. Vorbemerkung

Das bei der DORNIER-System GmbH in Friedrichshafen im Auftrag des BMVg entwickelte PEARL-Übersetzungssystem wurde schon wiederholt der Öffentlichkeit vorgestellt, darunter in der PEARL-Rundschau (1) und auf der Prozeßrechnertagung 1981 (2,3). Die folgende Darstellung kann sich daher darauf beschränken, die für einen potentiellen Anwender des Systems interessanten Aspekte zusammenzufassen und entsprechend dem für die Beiträge zu diesem Heft der Rundschau vorgegebenen Schema darzustellen.

2. Implementierter Sprachumfang

Der Sprachumfang richtet sich, wie viele der Eigenschaften des DS-PEARL Übersetzungssystems, nach den Anforderungen der Haupteinsatzgebiete Luft- und Raumfahrt sowie der Wehrtechnik:

- Hohe Effizienz
- Vermeidung jeglichen Overheads
- Einsatz verteilter Systeme

Es wurden also diejenigen Sprachelemente aus Basis-PEARL nicht verwendet, bei denen Effizienzprobleme oder unnötiger Overhead zu erwarten waren.

Insbesondere sind dies:

- Filehandling (Bordrechner besitzen üblicherweise keine Hintergrundspeicher)
- Formatierung (Es gibt dort auch kaum Druckausgabe)
- Absolutzeit (Die Zeit wird praktisch immer relativ zum Missionsanfang gemessen)
- Signals (Unplanmäßige Softwarezustände dürfen im operationellen System nicht mehr auftreten)
- Strukturen (Es treten meist nur gleichartige Meßdaten auf)

Zur Programmierung verteilter Systeme mußten allerdings gewisse Ergänzungen an Basis-PEARL vorgenommen werden. Dabei sollten folgende Randbedingungen eingehalten werden:

- Möglichst geringe Änderungen gegenüber dem Originalzustand von PEARL
- "Strategiefreiheit" der Zusatzkonstrukte, d.h. der Anwender sollte in die Lage versetzt werden, das seinem Projekt angemessene Sicherheits- und Redundanzkonzept selbst zu formulieren.

Daraus ergaben sich im wesentlichen folgende Ergänzungen:

- Netzglobale Größen (Attribut: GLOBAL NET) der Arten: Task, Sema und Daten
- Operationen auf derartige Größen (aber ohne zusätzliche Sprachelemente; nur Semantikerweiterung der entsprechenden vorhandenen Operationen)

Darüberhinaus wurden noch Möglichkeiten zum Anschluß externer (z.B. in Assembler geschriebenen) Tasks und Prozeduren (EXTERNAL) und zum Ein- und Anschalten von Laufzeitüberwachungen (CHECK, NOCHECK) vorgesehen.

3. Verwendete Übersetzertechnologie

Die verwendete Übersetzertechnologie ist in den erwähnten Veröffentlichun-

gen bereits dargestellt, deshalb seien ihre wesentlichen Eigenschaften hier nur noch einmal kurz zusammengestellt:

- Adaptierbarkeit an neue Zielmaschinen durch die klassische Aufteilung in einen zielmaschinenunabhängigen oberen Computerteil und einen Codegenerator
- Adaptierbarkeit an verschiedene Übersetzungsmaschinen durch weitestgehende Erstellung des Übersetzungssystems in einer höheren Sprache. Aus Gründen der einfachen Verfügbarkeit und der Transparenz des Systems für den Kunden wurde Standard-FORTRAN gewählt.
- Offenheit des Gesamtsystems für neue Zielrechnerarchitekturen und Optimierungsalgorithmen durch Verwendung einer zielrechnerunabhängigen Zwischensprache, dem "Triplecode".

Bild 1, das (1) entnommen ist, gibt eine Überblick über Struktur und Komponenten des Übersetzungssystems.

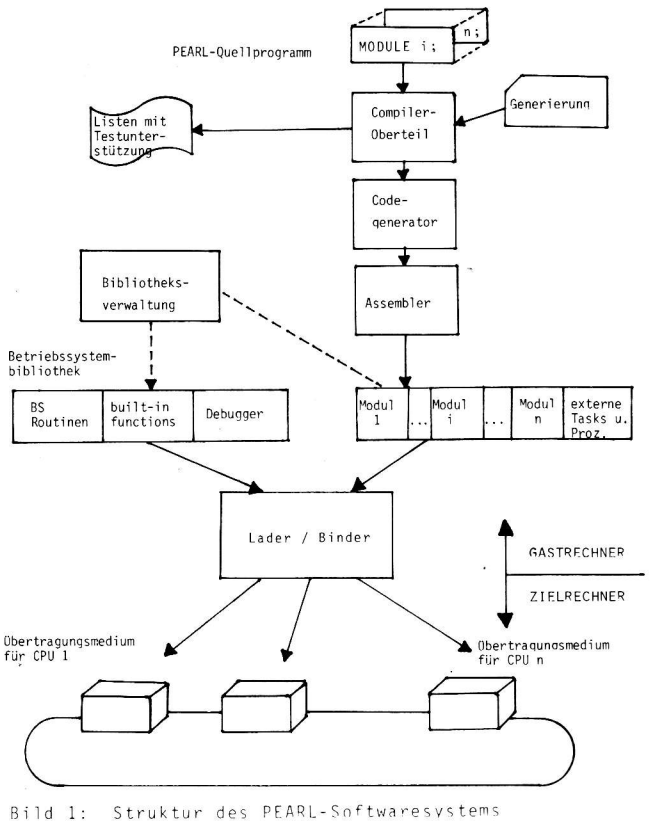

Bild 1: Struktur des PEARL-Softwaresystems

4. Komponenten des Übersetzungssystems

Aus der inzwischen allgemein verbreiteten Einsicht heraus, daß ein Compiler allein noch keine zufriedenstellende Unterstützung bei der Anwendung einer höheren Sprache darstellt, wurde das DS-PEARL-Übersetzungssystem so angelegt, daß es vor allem wirksame Unterstützung bei Test und Integration von Programmen bietet. Seine Komponenten werden im Folgenden kurz im Einzelnen dargestellt.

4.1 Oberer Compilerteil

Dieser übersetzt PEARL-Module in Tripelcode. Er erkennt ca. 200 verschiedene Fehler und identifiziert sie mittels Anweisungsnummer, Objektname und ausführlicher Zusatzangaben. Während der Übersetzung können folgende Listen erzeugt werden

- Quellprogramm
- Cross-Referenz-Listen für folgende Objekte:
  - Variable
  - Tasks
  - Semaphore
  - Proceduren
  - Marken
  - Dations

  wobei deren jeweilige Attribute (z.B. GLOBAL) angegeben werden.
- Aufrufhierarchie
- Taskingverhalten
- Semaphoreverhalten
- Variablenlokation

Der Compiler führt außerdem eine Optimierung bezüglich Eliminierung gemeinsamer Teilausdrücke durch.

4.2 Codegenerator

Dieser erzeugt relativ adressierten symbolischen Assemblercode des jeweiligen Zielrechners. Dieses Verfahren hat sich als Testhilfe und vertrauensbildende Maßnahme sehr bewährt. Durch die Strategie des Codegenerators werden unnötige "STORE-LOAD-Paare" und dergl. automatisch vermieden und damit gut optimierter Code erzeugt.

4.3 Assembler

Dieser ist Bestandteil des Übersetzungssystems und wird, wenn möglich, aus der Unterstützungssoftware des Herstellers des Zielrechners übernommen. Wenn nötig, wird er jedoch adaptiert oder auch neu erstellt.

4.4 Pre-Linker

Falls der vom Hersteller des Zielrechners bereitgestellte Binder nicht die für die Montage von PEARL-Programmen notwendigen Leistungen bietet, wird ein Pre-Linker mitgeliefert, der folgende Manipulationen ermöglicht:

- Angabe der zu bindenden Module
- Verteilung des Codes auf RAM und ROM
- Verteilung auf verschiedene Prozessoren
- Überprüfung auf vollständige Definition von globalen Größen
- Hinzubinden aller notwendigen Betriebssystemroutinen
- Sortieren von Kontrollblöcken und Segmenten
- Erstellung der Kommandofolge für den ladenden Binder.

4.5 Ladender Binder

Dieser führt den eigentlichen Bindervorgang durch und erzeugt absolut adressierten Code. Falls er nicht aus der Herstellersoftware übernommen werden kann, wird er mitgeliefert und bildet dann mit dem "Pre-Linker" eine Einheit

4.6 Betriebssystembaukasten

Diese einzigartige Komponente des DS-PEARL-Übersetzungssystems ermöglicht einen effizienten Einsatz von PEARL auch bei kleinsten Zielsystemen. Das Betriebssystem bildet nämlich keinen monolithischen Block, sondern besteht aus einem Satz von Routinen, die je nach den Anforderungen des Anwendungsprogramms automatisch dazugebunden werden. Operationsbasis der Routinen sind die vom Compiler bereits in passender Anzahl erstellten Verwaltungslisten (Task-Control-Blocks, time-order-blocks, interrupt-blocks, etc.)

Der Minimalausbau des Betriebssystems (Kern) umfaßt je nach Architektur des Zielrechners 0,3 - 0,5 K 16-bit Worte. Dieser Kern umfaßt:

- Initialisierung
- Dispatcher (Taskmanager)
- Exitroutine (Kurzfassung des Dispatchers, die durchlaufen wird, wenn sicher ist, daß kein Taskwechsel erfolgen kann)

Dazu können dann je nach Bedarf folgende Bausteine kommen:

- Uhrverwaltung
- Interruptverwaltung
- Taskbeendigung (regulär)
- Taskbeendigung (irregulär) (TERMINATE)
- Taskunterbrechung (SUSPEND)
- Taskfortsetzung (CONTINUE)
- Taskaktivierung (AKTIVATE)
- Schedule löschen (PREVENT)
- Semaverwaltung (REQUEST, RELEASE)
- Bedienroutine
- Kommunikation
- Charakter-E/A (PUT und GET)
- Prozedureintritt/-austritt
- Feldelementadressrechnung
- Arithmetikroutinen für FLOAT und DUR
- Vergleichsroutinen für FLOAT und DUR
- Datentypwandlungsroutinen
- Standardfunktionen (ABS, SIGN)
- Laufzeitfehlerroutine

Das Betriebssystem im Vollausbau umfaßt je nach Architektur des Zielrechners maximal 4-6 K 16-bit Worte.

4.7 Bedien- und Testsystem

Dazu gehören zunächst einmal die bereits erwähnten auführlichen Listen, die dem Benutzer als Referenzdokumente bei der Durchführung von Test dienen.

Dazu kommen Laufzeitüberwachungen, die entweder in Form von zusätzlichem Code oder als spezielle Betriebssystemsroutinen in das Programm eingebaut werden. Folgende Fehler werden erkannt und gemeldet:

- Feldindexüberlauf
- Division durch Null
- Datenbereichsüberschreitung (range)
- Wandlungsfehler

Diese Laufzeitüberwachungen können durch CHECK/NOCHECK ein-, bzw. ausgeschaltet werden, wobei die Möglichkeit besteht, die Gültigkeit einer solchen Instrumentierung bis herunter auf die Ebene einer einzelnen PEARL-Anweisung zu begrenzen.

In den Code können auch eine Reihe von Traces eingebaut werden:

- Jump-Trace
- Subroutine-Trace
- Call-Trace
- Task-Trace

Eine zusätzliche Komponente ist der Debugger, der zusammen mit dem übersetzten Programm geladen werden kann. Er bietet folgende Testmöglichkeiten:

- Starten und Fortsetzen von Tasks
- Setzen und Löschen von Haltepunkten
- Ausgabe von Haltepunktinformation
- Eingeben und Anzeige von Werten
- Exit (=Übergang zum Normalablauf des Programmes)

Der Debugger ist in zwei Ausbaustufen konzipiert:

1. Fehlersuche auf Assemblerebene
2. Fehlersuche auf PEARL-Ebene

Derzeit fertiggestellt ist die erste Ausbaustufe.

4.8 PEARL-Bibliotheksverwaltung

Um auf die speziellen Eigenschaften des PEARL-Modulkonzeptes eingehen zu können und es dem Anwender zu ermöglichen, seine System- und BS-Bibliothek selbst zu erweitern, wird eine eigene Bibliotheksverwaltung mitgeliefert. Sie enthält folgende Funktionen:

- Einfügen eines neuen Moduls
- Löschen eines Moduls
- Ausgabe des Inhaltsverzeichnisses
- Ändern von Modulnamen

5. Übersetzungsrechner

Das DS-PEARL Übersetzungssystem ist auf allen Rechnern mit einem geeigneten FORTRAN-Compiler ablauffähig. Bisher wird es auf folgenden Rechnern eingesetzt:

- DEC PDP 11/70 oder 11/44 mit 64 K Worten und Platte (RK05, RL01, RK06, RK07)
- AEG-Telefunken 80-20/4 mit 64 K Worten und Platte (KPS 3721)
- SIEMENS 7760
- DEC PDP 10

6. Zielmaschinen

Das System wurde bisher an folgende Zielmaschinen adaptiert:

- DORNIER MUDAS Datenprozessor 432
- DORNIER MUDAS Datenprozessor 433
- AEG-Telefunken 80-20
- Intel 8086

7. Lieferumfang

Die Lieferform des Übersetzungssystems (Lochkarten, Platte, Band, Diskette, etc.) kann vom Kunden gewählt werden. Die Installierung auf dem Übersetzungsrechner erfolgt entweder durch DS oder (nach kurzer Einweisung) durch den Kunden selbst.
Als Arbeitsunterlage dient eine PEARL-Sprachbeschreibung und -handbuch.
Auf Wunsch werdern durch DS auch Schulungskurse durchgeführt.

(1) H.J. Schneider:
PEARL-Softwaresystem für gekoppelte Klein- und Mikrorechner;
PEARL-Rundschau, Band 1, No.4, Dez. 1980 (S. 3-5)

(2) M. Amman:
PEARL für verteilte Systeme;
Vortrag auf der Fachtagung "Prozeßrechner 1981", März 1981, München

(3) F. Graf:
PEARL für Mikrocomputer;
Vortrag auf der Fachtagung "Prozeßrechner 1981", März 1981, München