P. HEINE, F.v. STÜLPNAGEL
Fraunhofer - Institut
für Informations- und Datenverarbeitung
IITB
Karlsruhe

PETSY: Ein PEARL TEST SYSTEM zum Austesten von PEARL-Programmen eines verteilten Systems

Auszug:

Im Rahmen eines Projektes mit dem Ziel, die höhere Echtzeitsprache PEARL in ein verteiltes Rechnersystem zu implementieren, wurde in Form eines PEARL-Programms das Testsystem PETSY für die S310 erstellt. Dieses ermöglicht, PEARL-Programme in einem beliebigen Rechner des Systems unter seiner Kontrolle ablaufen zu lassen. Dafür mußte der bestehende PEARL-Compiler und das PEARL-Betriebssystem um Schnittstellen zum Testsystem erweitert werden.

Inhaltsverzeichnis :

1. Einleitung

Mit der fortschreitenden Verbreitung der höheren Realzeitsprache PEARL bei der Programmierung automatisierter technischer Prozesse ergibt sich die Notwendigkeit, dem Entwickler mächtige Hilfsmittel für die Programmentwicklung zur Verfügung zu stellen.

Einen wesentlichen Bestandteil eines derartigen Systems von Programmproduktionsmitteln stellt ein Testsystem dar, das die Entwicklung komplexerer Programme unterstützt und damit erheblich zur Zeit- und Kostenersparnis bei der Automatisierung technischer Prozesse beiträgt.

Mit dem Entwurf eines PEARL-Testsystems wurde im Laufe des Jahres 1980 am Fraunhofer-Institut für Informations- und Datenverarbeitung (IITB) begonnen. Dabei konnte auf eine Studie aufgebaut werden, die anläßlich eines Referates für den PDV-Arbeitskreis 'Testen und Verifizieren von Prozeßrechnersoftware' erstellt wurde /2/ und in der Anforderungen an Testwerkzeuge formuliert sind.

Die Implementierung erfolgte auf der vorhandenen Hardwarekonfiguration eines fehlertoleranten Mehrrechnersystems. Dieses ist ausreichend veröffentlicht /1/, so daß hier anhand der Abb. 1 nur noch kurz darauf eingegangen werden soll:

Das verteilte Mehrrechner-System des IITB Karlsruhe ist eine ringförmige Konfiguration sogenannter RDC-Stationen, dabei besteht eine RDC-Station aus einem ringsteuernden Mikroprozessor und einem Prozeßrechner, auf dem AS10- sowie PEARL-Programme (ebenso wie auf den S310 Rechnern /3/, /4/) ablaufen können. Der Benutzer hat Zugriff zum Ring über einen S310-Rechner, an welchen Terminal, Bildschirm, Platte und EAF-System /5/ angeschlossen sind.

Bei der Implementation des PEARL-Testsystems PETSY auf dem RDC-System dient die S310 als 'host-Rechner', der den Monitor des PETSY enthält, unter dessen Kontrolle PEARL-Programme in de RDC-Satelliten-Rechnern unter seiner Kontrolle ablaufen können.

2. Kurze Charakterisierung der höheren Realzeitsprache PEARL

Auf die Sprache PEARL soll hier nur sehr kurz insoweit eingegangen werden, wie es für das Verständnis der Konfiguration und der Kommandosprache von PETSY nötig ist. Für weitergehendes Interesse sei auf die Referenzen /6/, /7/, /8/ verwiesen.

PEARL ist eine Abkürzung für 'Process and Experiment Automation Realtime Language'. Die Comiliereinheit eines PEARL-Programmes ist ein MODUL. Dieser kann TASKs, Unterprogramme oder Daten enthalten.

Dabei sind TASKs die konkurrierend ablauffähigen Prozesse der Sprache. Tasks können sich deshalb in verschiedenen Zuständen befinden. Zustandsänderungen werden durch die Anweisungen ACTIVATE, SUSPEND, CONTINUE, RESUME, TERMINATE herbeigeführt, die weitgehend selbsterklärend sind. Diese Zustandsänderungen können noch Bedingungen unterworfen werden, die erst den Ablauf einer vorgegebenen Zeit oder das Eintreffen eines Interrupts erfordern.

PEARL stellt SEMA- und BOLT-Variablen für die Synchronisation zur Verfügung, mit den möglichen Anweisungen REQUEST, RELEASE bzw. ENTER, LEAVE, RESERVE, FREE.

Interruptanweisungen sind ENABLE, DISABLE und TRIGGER.

3. Konfiguration des PETSY

Die Abb. 2 zeigt die Konfiguration des PETSY. Sie besteht aus mehreren PEARL-Modulen mit Schnittstellen zum Anwender, zum Compiler und zum Betriebssystem. Die Verbindung zwischen den Monitor-Modulen (die den Dialog mit dem Benutzer und die Kommandointerpretation beinhalten) und dem Testling wird durch zwei Kommunikationsmodule hergestellt. Dabei befindet sich der eine stets im 'host-Rechner' (S310) zusammen mit dem Monitor und der andere auf einem der Satellitenrechner (RDC) zusammen mit dem Testling.

Um kurze Antwortzeiten, sowie eine gute Modularisierung zu erhalten, wurde die Gesamtaufgabe auf einzelne konkurrierende Tasks verteilt.

So enthält der Monitor eine Task für den Dialog mit dem Benutzer, eine Task für die Ausgabe auf Drucker und eine Task für den Empfang einer Nachricht, falls der Testling einen Haltepunkt erreicht hat.

Die Kommunikationsmodule enthalten je eine Task zum Senden und Empfangen von Nachrichten. Ferner enthält der dem Testling zugeordnete Kommunikationsmodul (entsprechend dem Monitor-Modul) noch eine Task für die asynchrone Unterbrechung, falls der Testling auf einen Haltepunkt aufläuft.

Ein PEARL-Betriebssystem (PBS) und ein Netzwerk-Betriebssystem (NBS) unterstützen auf unterster Stufe den konkurrierenden Ablauf der PEARL-Task bzw. den Nachrichtenverkehr.

Diejenigen Tasks, die stets zusammen im gleichen Rechner liegen, synchronisieren sich untereinander über Synchronisationsvariable (Semaphoren); können die miteinander kommunizierenden Tasks in verschiedenen Rechnern liegen, so erfolgt die Synchronisation über den Austausch von Nachrichten mit time-out. Dadurch ist die Konfiguration des PETSY unabhängig davon, ob sich Monitor und Testling auf einem Rechner oder auf verschiedenen Rechnern befinden.

4. Schnittstellen

4.1 Anwender-Schnittstelle (Kommandosprache)

Die der Entwicklung des PEARL-Testsystems zugrundegelegte Anforderungsdefinition enthält einen Funktionsumfang, der für einen effektiven Test von Programmen, die in einer Echtzeitsprache formuliert sind, als unentbehrlich angesehen wird

und der auf eine, dem Niveau der höheren Programmiersprache angemessene, leichte Handhabbarkeit abgestimmt ist. Dabei wird zunächst auf komplexere Funktionen und über das Notwendigste hinausgehenden Komfort verzichtet.

Die Funktionen lassen sich grob in drei Gruppen gliedern. Zu einer ersten Gruppe lassen sich Funktionen zusammenfassen, die im wesentlichen für den zeitunkritischen Test eines Programmes geeignet sind und auch in den meisten anderen Test- und Debugprogrammen vorkommen.

Dazu gehören Funktionen zur Bestimmung von Testpunkten, Funktionen zum Anzeigen und Verändern von Speicherinhalten, sowie Funktionen zum Starten und Anhalten des Testlings.

Eine zweite Gruppe enthält Funktionen, die den Bediener des Testsystems unterstützen. Dazu gehören Hilfs- und Auskunftsfunktionen.

In einer dritten Funktionsgruppe lassen sich die Funktionen zusammenfassen, die die Besonderheiten der Echtzeitsprache PEARL, die im vorhergehenden Abschnitt kurz aufgeführt sind, berücksichtigen.

Mit diesen Testfunktionen kann unmittelbar in die Steuerung der konkurrierenden Prozesse des zu testenden Programmes eingegriffen werden, wobei die Zeit als Einflußgröße eine wesentliche Bedeutung hat. Bei diesen zeitkritischen Testanweisungen sind nicht die Objekte des Prozesses, sondern die des Betriebssystems Gegenstand der Betrachtung bzw. der Manipulation.

Hierher gehören Funktionen zur Prozeßverfolgung, Prozeß-Kontrollanweisungen, Systemübersichtsfunktionen sowie Funktionen zur Beeinflussung des Gesamtsystems.

Die Abb. 3 enthält alle in der Anforderungsdefinition festgelegten Funktionen zusammen mit den entsprechenden Kommandos.

4.2 Compiler-Schnittstelle

Die Schnittstelle des Compilers zum PETSY besteht pro Modul aus Listen, welche Informationen über die Bezeichner und ihre Attribute enthalten, sowie über Adressen und Zeilennummern. Für den in Abb. 3 aufgeführten Funktionsumfang werden 2 Listen benötigt. Dabei ist die erste Liste ein sogenanntes Vormerkbuch, bestehend aus :

- Bezeichner-Name
- Speicherklasse (global, lokal)
- TYP
- entsprechend dem TYP:
  Länge oder Genauigkeit oder Unterscheidungskennung
- Relativadresse zum Modulumfang

und die 2. Liste enthält :

- Quellzeile
- Relativadresse zum Modulumfang

Es bleibt noch zu bemerken, daß die Adressierung relativ zum Modulumfang zu Beginn des Monitor-Laufes dadurch absolutiert wird, daß der Monitor vom Betriebssystem eine Liste der Modul-Ladeadressen erhält.

4.3 Betriebssystem-Schnittstelle

Auf unterster Ebene liegt der 'Kern' des PETSY, welcher eine Erweiterung des bestehenden PEARL-Betriebssystems (PBS) bezügl. PETSY-Funktionen ist (Abb. 2). Auf eine der Funktionen des 'Kerns' soll an dieser Stelle etwas näher eingegangen werden, nämlich auf die Technik der Behandlung des Haltepunktes:

Der Inhalt der Adresse, an der ein Haltepunkt eingesetzt werden soll, wird weggespeichert und durch einen nicht interpretierbaren Befehl ersetzt. Läuft der Testling auf diesen Befehl, so erfolgt an dieser Stelle ein Ebenenwechsel auf die 'Fehlerebene' Ø. Von dort aus wird die, im Kommunikationsmodul des Satellitenrechners vorhandene, Haltepunkt-Empfangs-Task

aktiviert, welche dann die Adresse des Testpunktes zusammen mit der Nummer der Rechnerstation aus dem 'Kern' des PETSY abholt und dem Monitor zusendet.

5. Implementation

Der im IITB installierte PEARL-Compiler wurde von dem Softwarehaus W. Werum, Lüneburg zur Verfügung gestellt. Er läuft im IITB auf einer SR30/S310 und erzeugt wahlweise Code für den Ablauf auf SR30- oder S310/RDC-Maschinen. Der Sprachumfang des Compilers geht über BASIS-PEARL hinaus; sein genauer Umfang kann aus /7/ entnommen werden.

Das PEARL-Betriebssystem für S310/RDC wurde im IITB erstellt /4/.

PETSY wurde bisher nur in den Funktionsstufen I und II (vgl. Abb.3) realisiert, noch nicht in den Funktionen, welche die vom PEARL-Betriebssystem verwendeten Objekte betreffen (Stufe III); weitere vorläufige Einschränkungen sind, daß bisher nur auf Objekte zugegriffen werden kann, die auf Modulebene bekannt sind, ferner nur auf nicht indizierte Größen und bei Strukturen nur auf deren Namen, nicht aber auf die einzelnen Elemente.

Eine ernsthafte Problematik für eine vollständige Implementation des PETSY auf der S310 könnte nur der große Bedarf an Speicherplatz sein, da für den bisherigen Funktionsumfang schon bereits stark optimiert werden mußte.

Neben der Anwendung einer Overlay-Technik auf der S310, die in diesem Falle allerdings selbst erstellte werden müßte, wäre eine vollständige Implementation auf der SR30 eine Lösung. Dabei könnte weitgehend von der Portabilität des PETSY Gebrauch gemacht werden. Neu zu erstellen (wenn auch nicht neu zu definieren) wäre dabei die, unter Abschnitt 4.3 aufgeführte, Betriebssystem-Schnittstelle.

6. Zusammenfassung

Wir haben das PEARL-Testsystem PETSY vorgestellt, das im IITB Karlsruhe entwickelt wurde. Als Zusammenfassung sollen abschließend charakteristische Merkmale gesammelt aufgeführt werden:

- PETSY erlaubt es, PEARL-Programme auf der (PEARL-)Sprachebene als Testling auszutesten, unter weitgehender Verwendung der PEARL-Schlüsselwörter in der PETSY-Kommandosprache;
- dabei ist es modular derart konfiguriert, daß ohne Änderung der PETSY-Module ein Testling wahlweise auf dem S310-'host'-Rechner oder auf einem RDC-'Satelliten'-Rechner des zugrundeliegenden verteilten Systems ablaufen kann.
- Der modulare Aufbau von PETSY erlaubt ferner leicht Erweiterungen hinzuzufügen.
- PETSY besitzt dadurch, daß es in PEARL für PEARL-Programme geschrieben ist, den hohen Grad an Portabilität, wie die Sprachimplementation selbst.
- Durch die Methode des dynamischen Einsetzens der Testpunkte wird das Realzeit-Verhalten des Testling nur geringfügig geändert, und dieser benötigt keine spezielle Behandlung durch den Compiler, um im Test ablauffähig zu sein.

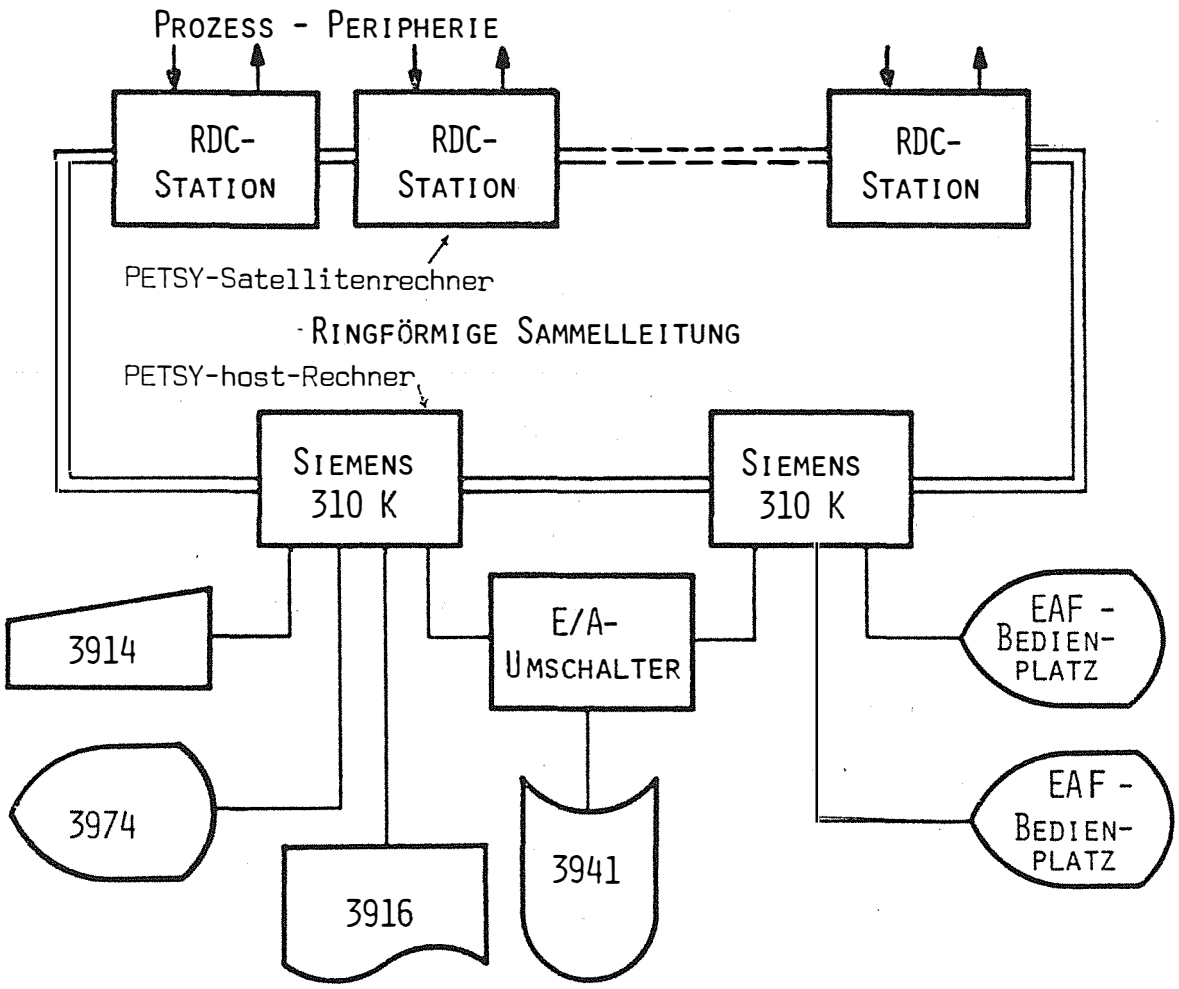
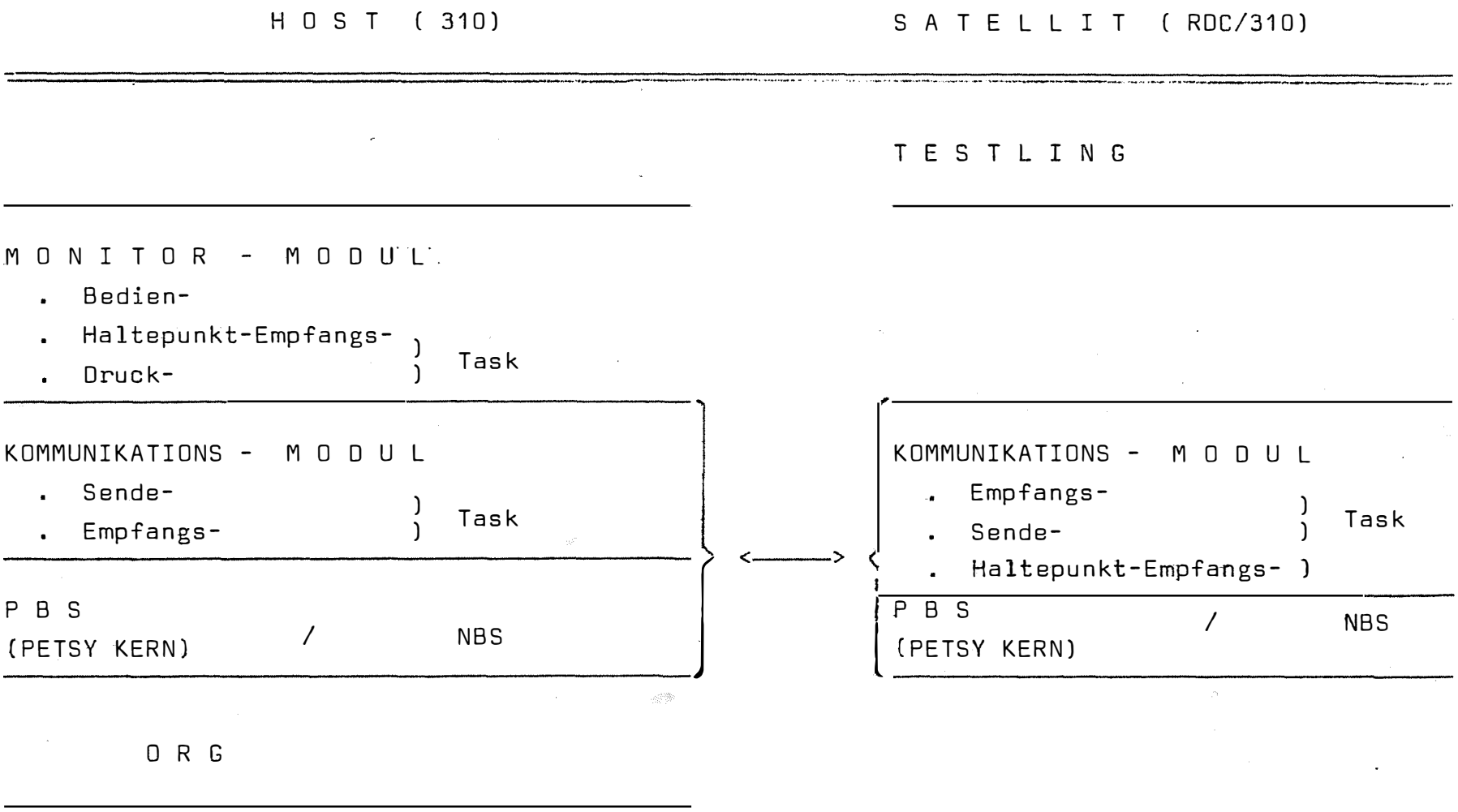


BILD 1: KONFIGURATION DES RDC-SYSTEMS

Fraunhofer-Institut für Informations- und Datenverarbeitung · IITB



Konfiguration des PETSY

Abb. 2

Funktion	Kommando	
Default Testobjekt	TEST	
Testpunkt setzen	BREAK	
Testpunkt löschen	DELETE	
Testpunkt aktivieren	INSERT	
Testpunkt deaktivieren	CANCEL	
Starten des Testobjektes	START	I
Weiterstarten	GO	
Test beenden	STOP	
Genereller Stop	KILL	
Anzeigen Speicherinhalt	DISPLAY	
Verändern Speicherinhalt	SET ...=...	
	SET ... IDENT	
Anzeigen Semavariable	DISPLAY	
Verändern Semavariable	REQUEST	
	RELEASE	
Anzeigen Boltvariable	DISPLAY	
Verändern Boltvariable	ENTER	
	LEAVE	
	RESERVE	
	FREE	
Anzeigen Taskzustand	DISPLAY	
Verändern Taskzustand	ACTIVATE	
	CONTINUE	III
	PREVENT	
	RESUME	
	SUSPEND	
	TERMINATE	
Task blockieren	LOCK	
Task entblockieren	UNLOCK	
Anzeigen von Interrupts	DISPLAY	
Interruptoperationen	TRIGGER	
	ENABLE	
	DISABLE	
Auflisten Testpunkte	LIST	II
Hilfsfunktion	HELP	
Verfolgen Ablauf	PATH-TRACE	IV
Verfolgen Variable	VALUE-TRACE	

PETSY-Kommandos
Abb. 3

Literaturverzeichnis

/1/ Echtzeitrechnersystem mit verteilten Mikroprozessoren, BMFT-Forschungsbericht DV 79-01

/2/ I. Hertlin, M. Mackert: Beitrag zu "Testen und Verifizieren von Prozeßrechner-Software", PDV-Bericht KfK-PDV 179, Dez. 79, des Kernforschungszentrums Karlsruhe

/3/ I. Hertlin, L. Lorenz: PEARL-Programmierung auf der Siemens 310: Übersetzungssystem, Beitrag zur SAK-Tagung 12.-14. Mai 80, KfK Jülich

/4/ G. Bonn, P. Heine: PEARL-Programmierung auf der Siemens 310: Ein PEARL-Betriebssystem, Beitrag zur SAK-Tagung 12.-14. Mai 80 KfK Jülich

/5/ Bildprogrammierbares Ein-/Ausgabe Farbbildschirmsystem (EAF) als Warte, PDV-Bericht 137, 1978, des Kernforschungszentrums Karlsruhe

/6/ Full PEARL Language Description, PDV-Bericht 130, 1977, des Kernforschungszentrums Karlsruhe

/7/ W. Werum, H. Windauer: PEARL, Vieweg-Verlag 1978

/8/ H.U. Steusloff: Programming Distributed Computer Systems with Higher Level Languages. IFAC Distributed Computer Control Systems, 1980.