PEARL – spezifische Rechnerkopplung

G. Landsberg, H. Meyerhoff, Bremen

1. Zusammenfassung

Ausgehend von dem Ein-/Ausgabe-Konzept der Realzeitsprache PEARL in Form von DATIONs werden Probleme bei der Implementierung bezüglich Universalität und Effizienz beschrieben.
In einem konkreten Beispiel einer Rechnerkopplung wird eine Lösung beschrieben, bei der die Forderungen nach höchster Effizienz ohne Umgehung des PEARL-Sprachumfangs realisiert wurde:
Die PEARL-Dation wurde weitgehend in Hardware abgebildet.

2. Einleitung

Die Programmiersprache PEARL (Process and Experiment Real-time Language) bietet im Gegensatz zu anderen Sprachen [1; 2; 3;] im Sprachumfang festgelegte Möglichkeiten für Prozeß-Ein-/Ausgabe.
Der wichtigste Begriff ist hierfür die Datenstation (DATION), ein virtuelles Gerät oder Kanal, das mit Attributen beschrieben wird (z. B. IN, OUT, DIM, INTERRUPT usw.) und auf das Anweisungen (z. B. TAKE, SEND usw.) ausgeführt werden können.

Bei vorhandener Hardware und einer dafür implementierten PEARL-E/A bietet das für den Prozeß-Programmierer einen hohen Komfort und eine leichte und schnelle Programmierung bei relativer Fehlerfreiheit.
Die Probleme werden - wie es auch sinnvoll ist - auf den PEARL-Implementator verlagert, der das Kunststück fertigbringen muß, einen festgelegten Einheitshut über den ausgedehnten Bereich der Prozeß-Ein-/Ausgabe der verschiedensten Prozesse zu stülpen.

Der wesentliche Gegensatz besteht dabei zwischen den Forderungen

- universeller Einsatz des festgelegten E/A-Konzeptes für alle Prozeßanwendungen und
- höchste Effizienz bezüglich der zeitlichen Ausführung und der Rechnerbelastung in besonderen Fällen.

Die Lösung des Konfliktes kann nur ein Kompromiß sein, der dann tragfähig ist und für das festgelegte Sprachkonzept in PEARL spricht, wenn z. B. 80% der Anforderungen mit der realisierten Implementierung abgedeckt werden kann.

Die übliche Methode, die restlichen 20% abzudecken, ist die Einführung von Assembler-Unterprogrammen in einem PEARL-Modul.

In einem konkreten Anwendungsfall wird geschildert, wie ohne Verlassen des PEARL-Sprachraumes durch weitgehende Implementation einer DATION in Hardware die Forderungen nach zeitlicher Effizienz bei geringer Rechnerbelastung realisiert wurden.

Zum besseren Verständnis wird vorher das DATION-Konzept in PEARL geschildert und auf die bei der Implementation auftretenden Probleme eingegangen.

3. Datenstationen

Datenstationen haben die Aufgabe, Information zwischen dem Arbeitsspeicher des Systems und der Umgebung zu übertragen. Ihr grundsätzlicher Aufbau ist in Bild 1 dargestellt.

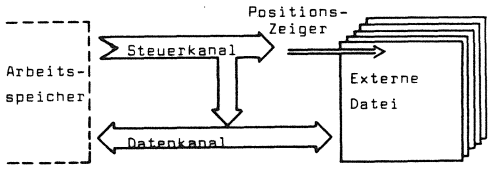

Bild 1 Bestandteile einer Datenstation

Vier Teile jeder Datenstation sind für den Programmierer von Interesse:

- Der Datenkanal
  Der Datenkanal bewirkt die eigentliche Informationsübertragung zwischen dem Arbeitsspeicher und der Umgebung (der externen Datei). Gewöhnlich werden die zu übertragenden Daten zunächst im Datenkanal zwischengespeichert und dann blockweise übertragen. Die Zwischenspeicherung kann sich gelegentlich auf die Funktionsweise auswirken, etwa bei einem Wechsel der Übertragungsrichtung.

- Der Steuerkanal
  Der Steuerkanal bewirkt die Steuerung des Betriebs der Datenstation entsprechend den Anforderungen des Programms. Er beeinflußt die Umwandlung der Daten zwischen ihrer externen Form und ihrer internen Form (der Form, in der sie im System gespeichert werden) und er steuert den Positionszeiger, der auf ein Element der externen Datei zeigt.

- Die externe Datei
  Eine Datenstation hat die Aufgabe, Daten zwischen dem Arbeitsspeicher des Systems und der Umwelt zu übertragen. Zum Verständnis des Konzepts der Datenstation ist es hilfreich, sich vorzustellen, daß die Daten außerhalb des Systems eine externe Datei bilden, so daß die Übertragung zwischen dem Arbeitsspeicher und der externen Datei stattfindet.

  Bei gewissen Datenstationen ist die Vorstellung, daß zur Datenstation eine externe Datei gehört, unmittelbar einleuchtend; beim Schnelldrucker z. B. besteht diese externe Datei aus dem bedruckten Papier, das den Drucker verläßt. Bei anderen Datenstationen existiert die externe Datei ebenfalls, allerdings nicht immer in einem so konkreten Sinne.

- Der Positionszeiger
  Dieser Zeiger gibt an, welche Position innerhalb der externen Datei von der Übertragung des nächsten Elements betroffen sein wird. Wenn der nächste Übertragungsvorgang eine Eingabe ist, zeigt der Zeiger auf dasjenige Element der externen Datei, das als nächstes gelesen wird, und wenn der nächste Übertragungsvorgang eine Ausgabe ist, zeigt der Zeiger auf diejenige Position in der externen Datei, auf die das nächste auszugebende Element geschrieben wird.
  Bei jeder Übertragung eines Elements schaltet der Zeiger automatisch um ein Element weiter (implizite Positionierung), so daß die Elemente der externen Datei ohne weiteres Zutun in fortlaufender Reihenfolge gelesen oder geschrieben werden können. Außerdem ist es möglich, mit Übertragungsanweisungen die Position, auf die der Zeiger weist, willkürlich zu verändern (explizite Positionierung) und so beim Schreiben oder Lesen von der natürlichen Reihenfolge der Dateielemente abzuweichen.

Grundlage jeder Ein- und Ausgabe sind die vorhandenen Geräte des Systems. Sie werden dem PEARL-Programm dadurch zugänglich gemacht, daß gewisse Datenstationen, die diesen Geräten unmittelbar entsprechen, dem System a priori bekannt sind. Die Namen und Eigenschaften dieser Datenstationen sind vollständig implementationsabhängig. Im Systemteil des PEARL-Programms werden die systemdefinierten Datenstationen in das PEARL-Programm eingeführt, indem der a priori bekannte Systemname der betreffenden Datenstation mit einem oder mehreren frei gewählten Benutzernamen assoziiert wird. In den Problemteilen können dann die Datenstationen unter ihren Benutzernamen angesprochen werden, nachdem sie jeweils spezifiziert worden sind.

Die Art der Spezifikation muß sich nach den Gelegenheiten der Hardware richten; dies ist

im einzelnen im Implementationshandbuch angegeben [7] .

Basis-PEARL erlaubt es dem Programmierer darüber hinaus, neue Datenstationen zu definieren; diese müssen über Datenwege mit den Datenstationen des Systems verbunden werden, da in Wirklichkeit natürlich Ein- und Ausgaben nur unter Verwendung der Hardware- und damit der Datenstationen des Systems - möglich sind. Die benutzerdefinierten Datenstationen können mit dem Attribut GLOBAL versehen werden und sind dann in allen Problemteilen des Programms nach entsprechender Spezifikation ansprechbar. Einzelheiten sh.[7].
Der Umgang mit Datenstationen ist für den Programmierer anfangs ungewohnt und für den Vorgang der Deklaration und der Eröffnung nicht ganz einfach. Das liegt u. a. darin, daß hiermit der Bezug zur speziellen Hardware des Systems hergestellt wird.
Die Anwendung im Problemteil ist dann jedoch einfach.
Eine Anweisung wie
SEND BYTE TO LAMPENFELD
ist schon Umgangssprache und leicht verständlich.

Warum die Ausführung obiger Anweisung dann aber ca. 1 ms dauert, ist vielen "Assemblerprogrammierern", die eine Ausgabe in 10 Assemblerbefehlen hinschreiben, restlos unklar.

Dazu muß man bedenken, daß die universelle DATION viele Sonderfälle abfangen muß. Im Laufzeitpaket für die DATION werden z. B. folgende Funktionen abgewickelt:

- Aufbereitung von Parametern und Adressen für die interne Schnittstelle.
- Überprüfung der Gültigkeit der Operation (z. B. ist die DATION vorhanden und eröffnet).
- Zugriffserlaubnis: Darf die aufrufende Task eines Laufbereiches (Users) auf die DATION zugreifen.
- Einstellung der DATION auf die Transferrichtung.
- Transfer des Objektes in den DATION-Buffer und evtl. Blockung der Daten.
- Status-und Fehleraufbereitung und Rücksprung.

Die genannte Zeit von 1 ms ist für die meisten Anwendungen ausreichend. Es muß jedoch berücksichtigt werden, daß für 1 ms auch Rechenzeit benötigt wird. Es kann in Ausnahmefällen Anwendungen geben, bei denen die Dauer der Ausführung und/oder die Rechnerbelastung mit den Anforderungen nicht in Einklang zu bringen sind. Von einem derartigen Fall wird im folgenden berichtet.

4. Das Projekt " BON "

Bei dem "BON - System" handelt es sich um eine zentrale Leitwarte für ein Nahverkehrssystem (sh. Bild 2).

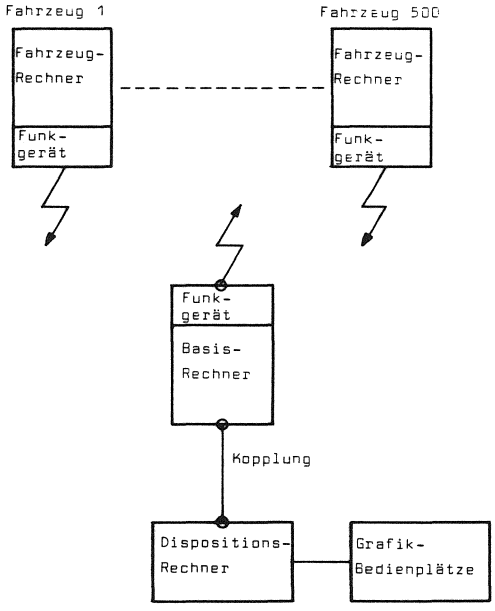

Bild 2: Leitsystem Nahverkehr (BON)

Über ein Funkinterface sind bis zu 500 Rechner in Fahrzeugen (Busse, Bahnen) mit der Leitwarte gekoppelt.
Die Aufgaben der Leitwarte werden auf dem Basisrechner (Aufgaben z. B. Fahrzeugaufrufe, Standortverfolgung) und auf dem Dispositionsrechner (Aufgaben z. B. Fahrplan Soll-Ist-Vergleich, Statistik) verteilt. Die Bedienung und die Ergebnisdarstellung (z. B. Standortdarstellung der Fahrzeuge) erfolgt auf Farb-Bildschirmgeräten.

Die bezüglich der Effizienz und der Rechnerbelastung kritischen E/A-Schnittstellen sind:

1. Schnittstelle des Basisrechners zum Funkinterface:
   Sämtliche Informationen von und zu den Fahrzeugrechnern werden über dicht gepackte ASCII-Telegramme der verschiedensten Formen abgewickelt (Funkbetrieb und 2400 Baud über 2 Schnittstellen). 50 Telegramme pro Sekunde müssen bei 30% Rechnerauslastung verarbeitet werden können.

2. Schnittstelle des Basisrechners zum Dispositionsrechner (Rechnerkopplung). Der Dispositionsrechner benötigt für seine Aufgaben einen schnellen Zugriff zu dem Datenmodell des Basisrechners (und umgekehrt). Die Datenkopplung muß schnell und ohne wesentliche Belastung des Basisrechners erfolgen.

Die Software des Projekts sollte in PEARL erstellt werden. In einer Vorstudie wurde festgestellt, daß für die beiden kritischen Schnittstellen die Forderungen unter Verwendung der Standard-DATION-Ein-/Ausgabe nicht erfüllt werden konnten.

Vom Lieferanten der Hardware und des PEARL-Systems - die Firma Krupp Atlas-Elektronik - wurde daraufhin für diese beiden Schnittstellen ein Verfahren vorgeschlagen und realisiert, das im folgenden am Beispiel der Rechnerkopplung ausführlicher geschildert wird.

5. Rechnerkopplung

Bild 3 zeigt die Hardwarekonfiguration.

Die eigentliche Parallel-Schnittstelle wird von einem programmierbaren E/A-Prozessor betrieben, der nach Initialisierung durch die CPU zum Speicher des Rechners Zugriff hat (per DMA, Direct-Memory-Access).

Ohne Mitwirkung der CPU werden Daten vom E/A-Prozessor aus dem Speicher geholt (oder zum Speicher gesendet), evtl. interpretiert und zur Parallelschnittstelle übertragen. Der Prozessor kann einen Hardware-Interrupt auslösen.

Das Verfahren besteht darin, die durch ein PEARL-Programm festgelegten strukturierten

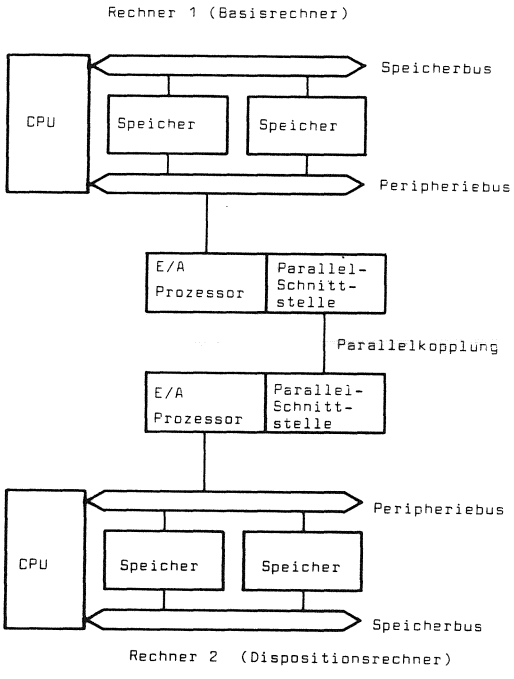

Bild 3: Hardware-Blockschaltbild, Rechnerkopplung

Daten nicht durch SEND- und TAKE-Anweisungen einer "DATION Parallelkopplung" zu transportieren, sondern diese Funktionen in den E/A-Prozessor zu verlegen.

Für den Fall der Datenkopplung von Rechner 2 (Dispositionsrechner) zum Rechner 1 (Basisrechner) werden die Hardwarefunktionen und die dazugehörigen PEARL-Anweisungen näher erläutert.

In einem globalen Datenmodul KOPLNG (Kopplung) werden die zu übertragenden Daten zusammen mit Steuerungsparametern durch Deklarationen aufbereitet (sh. Bild 5).

Der Block der Steuerungsparameter besteht jeweils aus drei Worten:

LNG, die Länge des Datenblocks

AUFT, eine gesetzte 1 bedeutet Start der Übertragung

SEMA, Die Nummern des globalen Semaphor des Partnerrechners, auf die nach Übertragungsende ein RELEASE erfolgen soll, womit ein Auswerteprogramm fortgesetzt wird.

Dem E/A-Prozessor wurde in der Initialisierungs-

phase des Gesamtsystems die Anfangsadresse des globalen Datenmoduls mitgeteilt. Die am Anfang des jeweiligen Datenblocks stehende Länge des Datenblocks gibt ihm die Informationen, um zyklisch das Auftragsbit AUFTR abzufragen.

Bei Vorliegen einer 1 werden die Daten einschließlich der Parameter zum anderen Rechner über die Parallelschnittstelle übertragen und die Daten dort in einem völlig symmetrisch aufgebauten Datensegment abgespeichert. Es kann nach Abspeicherung ein Interrupt ausgelöst werden. Daraufhin liest ein Treiber die mit übertragene Nummer des Semaphor aus. Durch ein RELEASE des Semaphor wird das aufgerufene Programm auf dem Rechner 2 fortgesetzt. Wenn der E/A-Prozessor eine Längenangabe 0 vorfindet, beginnt er seinen Poll-Zyklus wieder bei der Anfangsadresse (sh. Bild 5). Die Software-Struktur zeigt Bild 4.

Ein Modul FUB (Funkbedienung) will das Modul FZA (Fahrzeugaufruf) ansprechen. Da es sich in einem anderen Rechner befindet, wird ein Modul FZAKOP (Dummy Fahrzeugaufruf) angesprochen. Dieses Dummy-Modul startet die Kopplung. Die Parameter und die Daten werden mittels der Parallelkopplung übertragen. Auf der anderen Seite wird ein Auftragsmonitor angesprochen, der den eigentlichen Fahrzeugaufruf aktiviert.

Diese Struktur mit dem Dummy-Modul wurde gewählt, um eine Software-Struktur zu erhalten, bei der ohne

```
MODUL     KOPLNG     GLOBAL (1);
PROZEDUR;
.
.
.
.
.
.
DCL       LNG1       FIXED     GLOBAL    INIT(20);
DCL       AUFTR1     BIT(1)    GLOBAL    INIT ('0'B);
DCL       SEMA1      FIXED     GLOBAL;
DCL       DATEN1 (20) FIXED    GLOBAL;
.
.
.
DCL       POLL       FIXED     INIT (0);
MODEND;
```

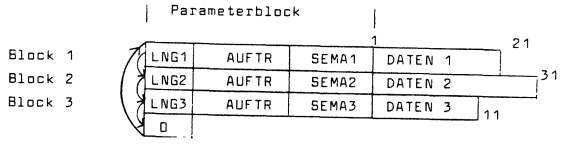

Bild 5 Speicherabbild der im PEARL-Modul KOPLNG (Kopplung) aufbereiteten Daten

große Änderungen die Module FUB (Funkbedienung) und FZA (Fahrzeugaufruf) auch auf einem Rechner ablaufen könnten.
Für das Modul FUB ist es "unsichtbar", ob das Modul FZA auf dem eigenen oder auf dem gekoppelten Rechner abläuft.

Ein vereinfachtes PEARL-Programm ist in Bild 6 und Bild 7 dargestellt.

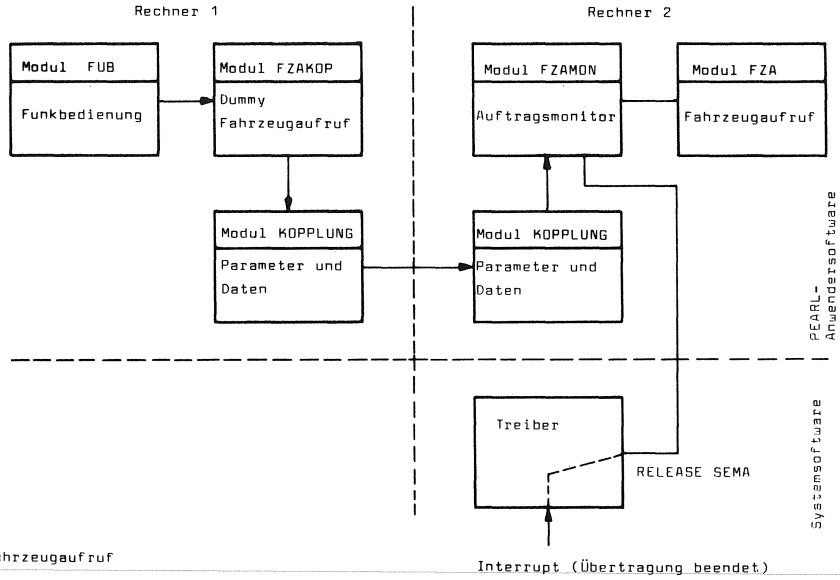

Bild 4: Struktur des PEARL-Systems für Rechnerkopplung: Funkbedienung bewirkt Fahrzeugaufruf

```
MODULE FUB;
PROBLEM;
SPC SCHNITTSTELLE ENTRY (FIXED) RESIDENT GLOBAL;
.
.
.
CALL SCHNITTSTELLE (E);    /* Schnittstellenaufruf */
.
.
.
MODEND;


MODULE FZAKOP;
PROBLEM;
.
.
.
SPC      AUFTR1      BIT(1);
SPC      DATEN1(20)  FIXED;
DCL      START       INV        BIT(1)  INIT ('1'B);

SCHNITTSTELLE: PROC (E FIXED) RESIDENT GLOBAL;
DATEN1 (1):= E;      /* Übergabe des Eingabeparameters */
AUFTR1:= START;      /* Start der Parallelkopplung */
END;
MODEND;
```

Bild 6 Vereinfachtes PEARL-Programm für die Rechnerkopplung, MODULE FUB (Funkbedienung) und FZAKOP (Dummy Fahrzeugaufruf) auf dem Rechner 2 (Dispositionsrechner)

```
MODULE FZAMON;
PROBLEM;
SPC SEMA1     SEMA GLOBAL;
SPC SCHNITTSTELLE ENTRY (FIXED) RESIDENT GLOBAL;
SPC DATEN1 (20) FIXED;

AUFTRAGSMONITOR: TASK PRIO1 RESIDENT GLOBAL;
REPEAT;
REQUEST SEMA1;  /* Warten auf Fertigwerden Parallelkopplung */
CALL SCHNITTSTELLE (DATEN1 (1)); /* Schnittstellenaufruf */
END;
MODEND;

MODULE FZA;
PROBLEM;
SPC DATEN1 (20) FIXED;
.
.
.
SCHNITTSTELLE: PROC (E FIXED) RESIDENT GLOBAL;
.
.
.
MODEND;
```

Bild 7 Vereinfachtes PEARL-Programm für die Rechnerkopplung, MODUL FZAMON (Fahrzeug-Monitor) und FZA (Fahrzeugaufruf) auf den Rechner 1/ Basisrechner

6. Ergebnis

Das Ergebnis des Verfahrens kann durch folgende Angaben charakterisiert werden:

- Ohne Verlassen des PEARL-Sprachraumes wurde ein effizientes Verfahren der Daten-Ein-/Ausgabe realisiert. Eine in Software implementierte DATION wurde umgangen.
- Pro Auftrag wird ein Grundaufwand von 100 µsec benötigt. Dazu ist u. a. der hier nicht beschriebene Aufwand für die Rückmeldung des erfolgreichen Transfers und die Datensicherung enthalten. Pro Datenwort sind 3,5 µsec für die Übertragung erforderlich. 170 µsec nach dem PEARL-Startbefehl (AUFTR1:=START, sh. Bild 6) steht der Datenblock von 20 Worten im Speicher des anderen Rechners.
  Die Zentraleinheit wird dabei so gut wie nicht belastet.
- Die Flexibilität einer Softwarelösung bleibt erhalten: Ohne Änderungen der Hardware können die Anzahl der Software-Koppelverbindungen und die Struktur der Daten verändert werden.
- Mit diesem Verfahren wurde gleichzeitig ein Beispiel gegeben, wie PEARL-Programme auf mehrere Rechner verteilt werden können.
- Das Ergebnis kann damit beschrieben werden, daß eine PEARL-DATION weitgehend in Hardware realisiert wurde.
  Eine komplette - für den Programmierer unsichtbare - Lösung einer Hardware-DATION kann durch folgende zusätzliche Maßnahmen erreicht werden:
  - o Die Aufbereitung der Parameterblöcke - in dem geschilderten Beispiel ein Teil des Anwenderprogramms - könnte der Kompilierer übernehmen.
  - o Ebenso könnte der Kompilierer die TAKE- und SEND-Anweisungen in entsprechende Start-Anweisungen für die Hardware umwandeln (z. B. SEND entspricht dem PEARL-Befehl nach Bild 6 in Modul FZAKOP (AUFTR1:=START).

7. Schlußbemerkung

Es wurde gezeigt, wie bei hohen Anforderungen an die Effizienz der Ein-/Ausgabe eines PEARL-Automatisierungssystems die Forderungen erfüllt werden können, ohne den PEARL-Sprachraum zu verlassen.

Es wurde ansatzweise die PEARL-Dation in Hardware realisiert.

Die gezeigte Struktur ist geeignet, PEARL-Programme auf mehrere Rechner zu verteilen.

Gleichzeitig ist die Realisierung ein Beispiel dafür, wie eine Hardware an die Struktur einer Software angepaßt werden kann.

Derartige Lösungen können nur bei enger Zusammenarbeit zwischen Systemhersteller und Anwender erreicht werden.
Der Lösungsansatz wird zur Nachahmung empfohlen.

8. Literatur

[1] P. Elzer:
Kurzüberblick über Entwicklung und Eigenschaften von PEARL im Vergleich mit anderen Programmiersprachen.
PEARL Rundschau Band 2 Nr. 2 1981

[2] A. Schwald, R. Baumann:
PEARL im Vergleich mit anderen Echtzeitsprachen.
PEARL Rundschau Band 2 Nr. 2 1981

[3] H. Sandmayr:
Sprachen für Echtzeitprogrammierung.
Fachtagung Prozeßrechner 1981, München.

[4] DIN 66253:
Teil 1 - Basic PEARL (Vornorm)
Teil 2 - Full PEARL (Normentwurf)

[5] B. Boysen:
PEARL auf einer 16 Bit-Rechenanlage mit Mehrbenutzersystem.
PEARL Rundschau Band 2 Nr. 6 1981

[6] M. Amann:
PEARL für verteilte Systeme.
Informatik-Fachberichte 39 1981,
Springer Verlag

[7] Krupp Atlas-Elektronik:
PEARL-Sprachbeschreibung BL 2052 A 179 4.82
PEARL-Implementationsbeschreibung BL 2052 A 166 8.81
PEARL Testmonitor BL 2052 A 158 6.81.