Kommunikation zwischen und mit PEARL - Echtzeitsystemen

Dipl.-Ing. Hans-Jörg Haubner, Karlsruhe

Zusammenfassung:

Ausgehend von einem einfachen Modell zur rechnerinternen Taskkommunikation werden prinzipielle Kommunikationsmechanismen mittels der Sprachmittel globale Daten und Semaphore beschrieben. Diese Mechanismen können mit wenigen zusätzlichen Betriebssystemfunktionen und -schnittstellen auf Echtzeitsysteme über lokale Netze (LAN) gekoppelt übertragen werden. Als Beispiel dient hier ein verteiltes Automatisierungssysytem.
Prinzipiell sind aber auch Kommunikationsfunktionen unmittelbar in geeigneten Hochsprachen realisierbar. THYNET ist ein Kommunikationssystem (WAN) für den betrieblichen Einsatz, bei dem von den Datenübertragungsprozeduren (X.25/Ebene 2) bis hin zum Netzwerkmanagement alle Funktionen in PEARL realisiert wurden.

An einem letzten Fallbeispiel wird die Einbettung eines PEARL-Systems in ein herstellerspezifisches Kommunikationssystem am Beispiel von SINEC demonstriert. Dieses System löst die Aufgabe, in einem Verbund von heterogenen Rechnern Daten, die lokal erfaßt werden, global allen Verbundteilnehmern mittels einfacher Verfahren für übergreifende Auswertungen konzentriert zur Verfügung zu stellen.

Stichworte: Verteilte Systeme, PEARL, von Kommunikationsfunktionen.

Abstract

Starting with a model for interprocess communication, basic communication mechanisms are described by means of the programming constructs, global-data and semaphores. Three approaches are presented providing communication functions for programming distributed systems. These are based upon functions integrated in the programming languages, functions realized in PEARL and the use of standard systems. In case studies for LAN and WAN, specific features and similarities of these appraches are explained.

Keywords: distributed systems, PEARL, programming of communication functions.

1. Ausgangssituation

Komplexe Prozeßautomatisierungs-Systeme werden heute schon beinahe selbstverständlich als verteilte Systeme ausgelegt. Ausschlaggebend hierfür sind Vorteile wie dedizierte Rechnerleistung vor Ort, Fehlertoleranz und Modularität.

Diesen Vorteilen stehen jedoch Aufwendungen für zusätzliche Kommunikations-Komponenten in Hard- und Software gegenüber. Üblicherweise werden dabei untere Schichten der Kommunikationsarchitektur durch Hard- bzw. Firmware-Komponenten, mittlere Schichten durch Betriebssystemfunktionen und/oder Kommunikationssysteme und obere Schichten durch Anwenderprogramme realisiert.

Allen Lösungen ist jedoch gemeinsam, daß entsprechend geeignete Schnittstellen und Funktionen dem Anwender zur Kommunikation zur Verfügung gestellt werden müssen. Dies kann auf der Basis vorhandener Sprachmittel der eingesetzten Programmiersprache oder durch Erweiterungen erfolgen.

Im vorliegenden Beitrag sollen für diese Vorgehenswseise verschiedene Lösungsansätze für PEARL-Programmsysteme einander gegenübergestellt werden. Als Fallbeispiele werden verteilte Prozeßautomatisierungssysteme, die am Institut für Informations- und Datenverarbeitung der FhG in PEARL programmiert und in der Industrie als Pilotanlagen eingesetzt wurden, herangezogen.

2. Ein einfaches PEARL-Modell der Kommunikation

Das Modell sieht zwei Instanzen, die durch PEARL-Tasks realisiert sind, vor, zwischen denen eine Kommunikation erfolgt. Die eine Task stellt den Sendeprozeß, die andere den zugehörigen Empfangsprozeß dar. Das Modell ist zunächst auf einen Speicher bzw. Adreßraum, den beide Prozesse gemeinsam benutzen, beschränkt.

Die Kommunikation kann dann wie folgt modelliert werden:

```
DCL PUFFER  TELE      GLOBAL;
DCL ANZEIGE QUITTUNG GLOBAL;
DCL ES  SEMA PRESET(0) GLOBAL;
DCL AS  SEMA PRESET(0) GLOBAL;

SENDER : TASK
         DCL X QUITTUNG;
         DCL Y TELE
          :
         PUFFER:= Y; Schreiben auf globale
                     Daten
         RELEASE ES; Synchronisation
                     Empfänger
         REQUEST AS; Warten auf Quittung
         X:= ANZEIGE; Quittung lesen

END;

EMPFANG : TASK;
          DCL Y TELE;
          DCL X QUITTUNG;
          REQUEST ES; Warten auf Empfang
          Y:= PUFFER; Lesen von globalen
                      Daten
          ANZEIGE:=X; Anzeige setzen
          RELEASE AS; Senden, Synchroni-
                      sieren
END;
```

Anhand dieses Modelles kann z.B. ein rechnerinternes Auftrags- oder Nachrichtensystem zur Interprozeßkommunikation realisiert werden. Solche Systeme werden als Software-Bus bezeichnet, da sie auf einfache Weise Interaktionen zwischen einer Vielzahl von Prozessen erlauben. Die dargestellten Abläufe sind dann als Sende- bzw. Empfangsprozedur zusammengefaßt und der Empfangspuffer durch ein Warteschlangensystem ersetzt.

3. Kommunikation in verteilten Systemen

Das vorgestellte Modell läßt sich auf verteilte Systeme mit getrennten Prozessoren und Speichern übertragen. Dazu müßte die Semantik das Sprachelement GLOBAL, in PEARL im Sinne von Rechner-global definiert, erweitert werden, etwa im Sinne von System-global. Zugriffe auf Datenobjekte, die als System-global deklariert sind und sich in einem anderen Rechner befinden, würden dann Kommunikationsvor-

gänge, die im Rahmen der PEARL-System-Umgebung zur Verfügung gestellt werden müßten, beinhalten.
Dieses Verfahren wird jedoch i.a. nicht praktiziert, üblich sind die beiden im folgenden vorgestellten Vorgehensweisen.

Zum einen besteht die Möglichkeit, die Sprache PEARL um entsprechende Kommunikations-Sprachmittel zu erweitern. Entsprechende Vorschläge hierzu liegen vor /1/, /2/. Ihnen gemeinsam ist, die beiden Elementaroperationen Senden und Empfangen als Sprachmittel ergänzt um Verbindungsstrukturen zur Verfügung zu stellen:

- TRANSMIT daten TO port;
- RECEIVE daten FROM port;

Diese entsprechen prinzipiell den Modellfunktionen, wobei z.T. die Synchronisationsmechanismen wie z.B. Warten, Nichtwarten und weitere Kontrollmechanismen variiert werden.

Die zweite Möglichkeit ähnelt stark der ersten; sie umgeht jedoch das Problem der Spracherweiterung, indem vergleichbare Funktionen auf der Basis von über CALL aufrufbaren Prozeduren zur Verfügung gestellt werden:

- CALL SENDF (port, daten,...);
- CALL RECEIV (port, daten,...);

Diese Funktionen wurden, wie im 1. Fallbeispiel dargestellt, in Rahmen eines verteilten Prozeßautomatisierungsystemes ins PEARL-Betriebssystem bzw. Laufzeitsystem integriert.
Beiden Lösungen gemeinsam ist, daß in der PEARL-Systemumgebung entsprechende unterlagerte Kommunikationskomponenten in Hard- und Software zur Verfügung gestellt werden.

4. Fallbeispiel: Verteiltes Prozeßautomatisierungssystem in einem Oxygenstahlwerk

Zur Automatisierung komplexer technischer Prozesse wurde im IITB ein verteiltes Prozeßautomatisierungssystem (RDC-System) entwickelt. Ein solches System wird seit 1981 von der Thyssen Stahl AG als Leitsystem zur Steuerung und Überwachung einzelner Verfahrensschritte im Oxygenstahlwerk eingesetzt /3/.

Die Konfiguration des Systems ist in Bild 1 dargestellt. Seine Strukturierung erfolgte bereichsweise anhand des vorgegebenen "natürlichen" Produktionsablaufes der zu automatisierenden Verfahrensschritte, Konverter und Argonspüle, Stahlentgasung, Sublanze, Stahlwerksrechner ergänzt um ein zentrales Wartensystem zur Bedienung der Gesamtanlage.

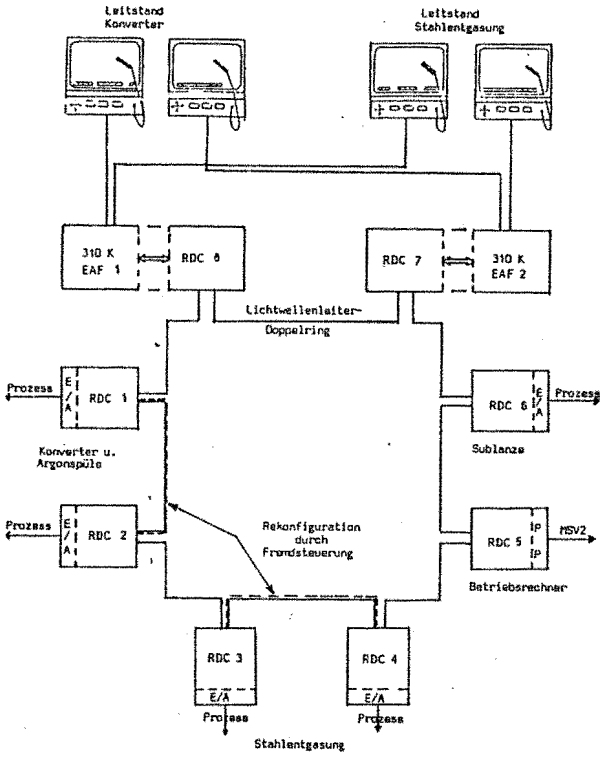

Bild 1. Verteiltes Leitsystem im Stahlwerk.

Die eigentlichen MSR-Aufgaben in den einzelnen Bereichen werden zunächst dediziert von Mikroprozessor-Systemen (Stationen), wahrgenommen. Diese sind untereinander und mit dem Wartensystem über ein lokales ringförmiges Kommunikationssystem auf der Basis eines Lichtwellenleiters miteinander verbunden.
Die Mikroprozessorstationen sind in PEARL programmiert. Als Kommunikationsfunktionen

werden SENDF, RECEIVE über das PEARL-Betriebssystem zur Verfügung gestellt, das hierzu um entsprechende Komponenten ergänzt wurde. Diese Kommunikationsfunktionen erlauben in der in Pkt. 3 beschriebenen Weise eine Kommunikation mittels Telegrammen bis maximal 128 Bytes. Auf diesen Basisfunktionen sind in PEARL geschriebene höhere Kommunikationfunktionen aufgesetzt. Es werden hiermit folgende typische Kommunikationsfunktionen wahrgenommen:

(1) Zyklische und ereignisgesteuerte Übertragung der lokal erfaßten Prozeßgrößen einschließlich Systemzuständen von den Stationen zum Wartensystem, zur Prozeßbeobachtung und -protokollierung.

(2) Übertragung von initialen Parametersätzen, Systemzuständen und Bedieneingriffen vom Wartensystem zu den Stationen zur zentralen Parametrierung und Bedienung.

(3) Zyklischer und ereignisgesteuerter Austausch von Parametersätzen, Prozeßgrößen und Systemzuständen paarweise zwischen den Stationen zur gegenseitigen Aktualisierung der Redundanzfunktionen.

Während die unter (1) und (2) beschriebenen Funktionen als "Fern-Meß- bzw. Steuerungs-Funktionen" aufgefaßt werden können, wird die unter (3) beschriebene Funktion dazu benutzt, die Stationen im Konverter- und Stahlentgasungsbereich paarweise in funktionsbeteiligter Redundanz arbeiten zu lassen. Fällt eine Station eines Paares aus, so übernimmt die Nachbarstation deren Aufgaben mit. Um hier eine weitgehend stoßfreie Übernahme zu erreichen, werden hierfür wichtige Daten wie z.B. Sollwerte u. Regelparameter von der prozeßführenden Station in der Nachbarstation ständig aktualisiert. Zur Steigerung der Effizienz wurden hier zwei weitere Basisfunktionen zur Kommunikation in das Betriebssystem integriert. Diese Funktionen HOLGM, ZUWGM ermöglichen es in einer Station direkt auf den Speicher einer anderen Station lesend bzw. schreibend zuzugreifen. Die eigentliche Empfangsfunktion entsprechend RECEIV entfällt. Zur Adressierung des "fremden" Speichers werden dabei einmalig im Anlauf der Stationen entsprechende Pointerlisten (Listen vom Typ REF) ausgetauscht, die sozusagen ein globales Adreßverzeichnis stationsübergreifender Datenobjekte darstellen.

Bei diesem System hat es sich gezeigt, daß sich selbst mit diesen einfachen Basisfunktionen komplexe Kommunikationsfunktionen aufbauen lassen. Insbesondere konnten die Nachteile verteilter Systeme, die insbesondere in einem erhöhten oft effizienzminderndem Kommunikationsaufwand und einer erschwerten oder unübersichtlichen zentralen Bedienung bestehen, vermieden werden.

5. Kommunikation mittels PEARL-Systemen

Bisher wurde gezeigt, wie Kommunikation zwischen PEARL-Systemen mit Basisfunktionen, die über die Systemumgebung zur Verfügung gestellt werden, realisiert wird. Es stellt sich die Frage, ob solche Funktionen auch mittels PEARL realisiert werden können. Dies soll am Beispiel der Ebene 2 von X.25, entsprechend HDLC, demonstriert werden.

Vorausgesetzt wird eine Hardware-Komponente am Rechner, die mechanische und elektrische Eigenschaften (Ebene 1) sowie die Parallel-Seriell-Wandlung der zu übertragenden Daten einschließlich der Flag- und CRC-Generierung und -Prüfung zur Verfügung stellt.

Diese Komponente kann über Ein- und Ausgabeoperationen von PEARL aus, betrieben werden. Hierfür sind in PEARL prinzipiell die Sprachmittel Systemteil, Dation, Read und Write vorhanden. Diese sind jedoch, wie im vorliegenden Fall, bei vielen Systemen auf Standardperipheriegeräte wie Terminal, Platte oder Prozeß-E/A beschränkt. Es ist daher sinnvoll, vergleichbare Funktionen als einfache Ein-Ausgabe-Operationen z.B. im Laufzeitsystem

zu installieren:

```
TYPE DATENBLOCK STRUCT [LAENGE FIXED,
                        DATEN(256) BIT(16)];
TYPE GERAET FIXED;
SPC ENTRY ATRANS (DATENBLOCK; GERAET);
SPC ENTRY ETRANS (DATENBLOCK, GERAET);
```

Weiterhin ist je ein Interrupt für Ein- bzw. Ausgabe notwendig, die bei den PEARL-Ein-Ausgabeoperationen implizit vorhanden sind:

```
SPC (I-IRUPT, O-IRUPT) INTERRUPT;
```

Damit können die beiden elementaren Funktionen Senden und Empfangen unter zu Hilfenahme zweier Task zur Interruptsteuerung analog zu dem im Pkt. 2 beschriebenen Modell wie folgt prinzipiell dargestellt, realisiert werden.

```
T1 : TASK;
     WHEN I-IRUPT CONTINUE;
     REPEAT;
        SUSPEND T1;
        RELEASE RECEIV-SEMA;
     END;
END;

T2 : TASK
     WHEN O-IRUPT CONTINUE;
     REPEAT;
        SUSPEND T2;
        RELEASE SEND-SEMA;
     END;
END;

T3 : TASK;
     :
        REQUEST RECEIV-SEMA; WARTEN
        CALL ETRANS (....);  LESEN
     :
END;

T4 : TASK;
     :
     CALL ATRANS (....); Schreiben
     REQUEST SEND-SEMA;  Warten
     :
END;
```

Anzeigen und Quittungen werden über entsprechende Protokollelemente von HDLC abgewickelt.

6. Fallbeispiel: THYNET, ein betriebliches Kommunikationssystem

THYNET ist ein am IITB entwickeltes paketvermittelndes Kommunikationssystem zum Aufbau von beliebig vermaschbaren Rechnernetzen. Das Gesamtsystem bildet einen heterogenen Verbund, der sich aus dem eigentlichen Kommunikationssystem und den daran angeschlossenen Teilnehmern zusammensetzt (s. Bild 2). Die Teilnehmer i.a. Rechner und Terminals sind heterogen und stellen die Datenquelle des Verbundes dar. Das Kommunikationssystem ist ein beliebig vermaschtes Netz aus Netzknotenrechnern, die über serielle Datenleitungen (Telefonleitungen) und Modems miteinander verbunden sind /4/.

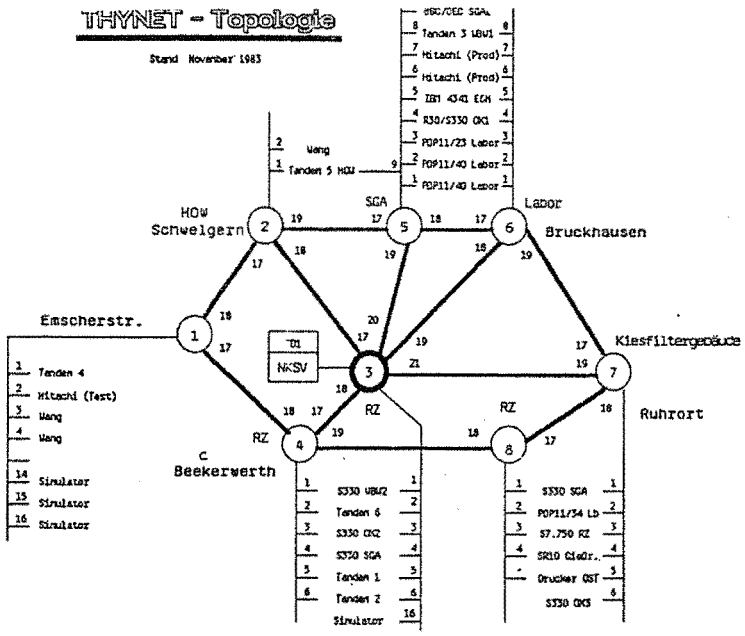

Bild 2: THYNET-Topologie.

Das Netz arbeitet nach dem Prinzip der Paketvermittlung. Übertragen werden Telegramme, deren maximale Länge 512 Bytes beträgt. Auf den Punkt-zu-Punkt-Verbindungen zwischen den Netzknoten wird einheitlich die Übertragungsprozedur HDLC eingesetzt. Die Leitungskontrolle wird dabei von sogenannten Prozedurprozessoren durch-

geführt. Diese Prozedurprozessoren werden vom PEARL-System über die beschriebenen Funktionen ATRANS und ETRANS angesteuert. Die Funktionen von HDLC werden bis auf FLAG- und CRC-Generierung über PEARL abgewickelt.

Die Ankopplung der Teilnehmer erfolgt ebenfalls über die Prozedurprozessoren. Die Netzzugangsprozedur ist mit der Ebene 3 von X.25 vergleichbar und ist ebenfalls in PEARL programmiert. Teilnehmer und Kontrolltrollinstanzen verkehren jeweils untereinander über Einzeltelegramme oder Sequenzen von Telegrammen, die über eine Wegesteuerung im Netz transparent übertragen werden.

Neben den bisher beschriebenen unbedingt notwendigen Funktionen zur Kommunikation zwischen Teilnehmern über das Netz sind im Netz weitere Funktionen zum System-Management integriert. Diese Leistungen stehen innerhalb des Netzes als Funktionsverbund zur Verfügung und können von speziellen Netzknoten aus netzweit benutzt werden. Alle Funktionen werden dabei über hierfür in den Netzknoten implementierte Kontrollinstanzen mittels spezieller Kontrolltelegramme abgewickelt.

Zum System-Management gehören folgende Teilsysteme:

- das Überwachungssystem zur zentralen Anzeige von Fehlern, Warnungen und Statusänderungen. Hiermit lassen sich z.B. Protokollereignisse und Leitungsfehler wie CRC-Fehler, Modemsignale, HDLC-Verbindungsaufbau überwachen, um schnell und zielgerichtet auf Störungen zu reagieren bzw. frühzeitig Fehlertrends zu erkennen.
- das Bediensystem zum zentralen Laden der Netzknoten (down-line-loading) und zur Umkonfigurierung und Umparametrierung von Netzknoten.
  Hiermit können im laufenden Betrieb z.B. einzelne Netzknoten nachgeladen werden bzw. bei Systemänderungen Konfigurationsparameter wie Wegetabellen modifiziert werden.
- das Ferndiagnosesystem zur Programmverfolgung (TRACE) und zum Speicherabzug (post-mortem-dump). Für Test- und Wartungszwecke lassen sich hiermit Programmabläufe zur Fehler-Lokalisierung analysieren.

Das beschriebene Programmsystem der Netzknoten ist vollständig in PEARL beschrieben. Es besteht aus 14 Modulen mit insgesamt 11 Task und benötigt ohne Betriebs- und Laufzeitsystem ca. 41 KW.

THYNET ist seit 1981 bei der Thyssen Stahl AG im Betrieb und wurde kontinuierlich auf den heutigen Umfang von 8 Netzknoten mit ca. 30 Teilnehmern erweitert.

Insbesondere bei diesem System kann gezeigt werden, daß auch Kommunikationsschichten der niedrigeren Ebenen wie z.B. Leitungssteuerung (HDLC) und Netzwerksteuerung (ähnlich X.25/3) sowie Netzwerk-Managementfunktionen mit PEARL effizient programmiert werden können.

7. Kommunikation über Standard-Kommunikationspakete

Als dritter Ansatz soll hier die Möglichkeit der Kommunikation über Standard-Programmsysteme, wie sie z.B. von Rechnerherstellern im Rahmen der Systemumgebung zur Verfügung gestellt werden, beschrieben werden.

Solche Standardsysteme besitzen i.a. Schnittstellen für Anwenderprogramme in Form von System-Makros oder Assembler-Prozeduren. Diese müssen auf PEARL-Niveau angehoben werden. Dies bedeutet, daß eine Schnittstellenschicht zu erstellen ist, die im wesentlichen aus Assemblerprozeduren besteht. Diese werden von PEARL aus mittels CALL aufgerufen und leisten die Umsetzung der Prozedurparameter von den PEARL- auf die Assembler-Konventionen sowie den Aufruf der System-Prozeduren. Damit sind darin die Kommunikationsdienste des Standardsystems von PEARL aus verfügbar.

Der Umfang dieser Dienste deckt notwen-

digerweise die unteren Ebenen 1 bis 4 des ISO Referenzmodelles ab, höhere Ebenen sind sehr stark herstellerspezifisch. So liegen auch einheitliche Schnittstellen nicht vor.

Am Beispiel von SINEC, einem Kommunikationssystem für die Rechnerfamilie SICOMP R und M, soll hier die grundlegende Vorgehensweise erläutert werden. Zur Verfügung stehen hier u.a. Dienste wie X.25 oder ein verbindungsorientiertes Transportprotokoll.
Die elementaren Kommunikationsfunktionen sind

- Anmelden des Anwendersubsystems (SNANS)
- Sendeaufruf (SNSDT)
- Empfangsaufruf (SNSREC)
- Warteaufruf für Quittung auf Senden bzw. Empfang (WAIT)

Diese Funktionen entsprechen den Basisfunktionen des Kommunikationsmodells. Damit ergibt sich eine dem Modell entsprechende Ablaufstruktur:

Senden:

```
CALL SNANS (...);  Anmelden
:
CALL SNSDT (...);  Senden
:
CALL WAIT  (...);  WARTEN
:
```

Empfang:

```
CALL SNANS (...);  Anmelden
:
CALL SNSREC(...);  Empfangsaufruf
CALL WAIT  (...);  Warten
Daten lesen
:
```

Für das eigentliche Lesen von Daten existiert keine spezielle Funktion. Bei allen Aufrufen werden entsprechende Anzeigen zurückgegeben, deren Auswertung hier nicht dargestellt wurde.

8. Fallbeispiel: Koppelrechnersystem für den Datenaustausch im heterogenen Rechnerverbund

Im Auftrag der Badenwerk AG wird derzeit im IITB ein Koppelrechnersystem entwickelt, das einen EVU-internen wie auch übergreifenden Datenaustausch zwischen heterogenen Leitstellen- und Energiewirtschaftsrechnern ermöglicht (s. Bild 3). Die Grundidee ist, über einen sog. Koppelrechner die einzelnen Teilnehmer des Verbundes sternförmig miteinander zu verbinden /5/.

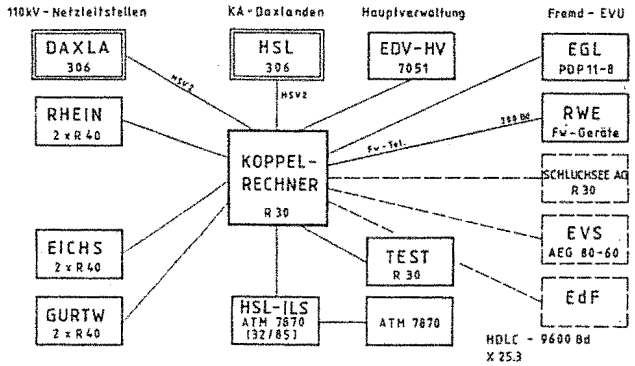

Bild 3: Konfiguration des KOPPELRECHNER-Systems.

Das System erfüllt folgende Hauptaufgaben:

- Für die Teilnehmer des Verbundes stellt der Koppelrechner ein Kommunikationssystem für den transparenten Transport von Daten zur Verfügung.

- Der Koppelrechner stellt für die Teilnehmer eine zentrale Datenbasis dar, in die bzw. aus der aktuelle Meßwerte, Zählwerte und Meldungen des elektrischen Netzes weitgehend wahlfrei von den Teilnehmern übernommen bzw. an die Teilnehmer übergeben werden.

Zur Erfüllung dieser Aufgaben wird SINEC M eingesetzt. Von SINEC werden die Subsysteme VS (Verbindungssteuerung, für Transportprotokoll), PV (Paketvermittlung für X.25), PU (Protokollunabhängige Kommunikation für einfache Kommunikation) und FV (Funktionsverbund) eingesetzt. Damit wird die Forderung nach Heterogenität weitgehend erfüllt.

Die Schnittstellen zu den Subsystemen liegen als Makros vor. Über eine in Assembler programmierte Schnittstellenschicht werden diese von PEARL aus angesprochen. Weiterhin werden die Protokoll-Unterschiede der unterschiedlichen Teilnehmeranschlüsse nivelliert, so daß der Verbund aus der Sicht eines einzelnen Teilnehmers homogen erscheint.

Bisherige Erfahrungen haben gezeigt, daß durch den Einsatz dieses Standardsubsystems weitgehende Hardware-Unabhängigkeit erreicht wird. Insbesondere können die Programme innerhalb der Systemfamilie einfach portiert werden.

9. Zusammenfassung und Ausblick

Die vorgestellten drei prinzipiellen Vorgehensweisen beruhen, wie gezeigt werden konnte, auf den gleichen Grundmechanismen des Kommunikationsmodells. Letzlich unterscheiden sie sich in der Anwenderoberfläche, die entweder von kommunikationsspezifischen Sprachmitteln eines erweiterten Sprachumfanges oder der allgemeinen Prozedurschnittstelle (CALL) gebildet wird. Bei der Konzipierung eines Systems ist die Wahl der richtigen Alternative daher stark beeinflußt von praktischen Überlegungen, wie Verfügbarkeit entsprechender Komponenten (Standardkommunikationssystem, Mehrrechner-PEARL, Betriebssystemfunktionen), Realisierungsaufwand, Effizienz, Portabilität und der zur Verfügung stehenden Hardware-Komponenten.
Weiterhin kann festgestellt werden, daß Architektur-Konzepte, wie z.B. durch das ISO-Referenzmodell für offene Systeme (OSI) definiert, auch im Bereich der Echtzeitsysteme ein geeignetes Strukturierungsmittel sind.
Von dem Standpunkt eines reinen Anwenders bietet es sich an, für die unteren Ebenen dieses Modells Standardkomponenten zu verwenden, um weitgehende Unabhängigkeit von Hardware- und Protokollnormen zu erreichen. Dabei kann es unter dem Aspekt der Vorteile höherer Programmiersprachen durchaus sinnvoll sein, solche Standards in PEARL zu realisieren. Daß und wie dies möglich ist, konnte am Fallbeispiel gezeigt werden.

Literatur

/1/ Fleischmann, A., et.al: Synchronisation verteilter Automatisierungsprogramme, Angewandte Informatik 7/83.

/2/Bügel, U., et.al: Mehrrechner-PEARL: Einsatzerfahrungen und Sprachvorschlag für Normung, FhG-Berichte 2-85.

/3/ Früchtenicht, H.W. et.al: Leitsystem zur Steuerung und Überwachung einzelner Verfahrensschritte in einem modernen Blasstahlwerk, Stahl u. Eisen (1982) Nr. 13.

/4/ Haubner, H., et.al.: THYNET ein Kommunikationssystem für den betrieblichen Einsatz, Informatik-Fachberichte (1985) 95.

/5/ Haubner, H., Räuber, H.: Datenaustausch im heterogenen Rechnerverbund I und II, Höhere Programmiersprachen in der Leittechnik 1984.

Dipl.-Ing. Hans-Jörg Haubner
Fraunhofer-Institut für Informations- und Datenverarbeitung
Sebastian-Kneipp-Str. 12/14
7500 Karlsruhe
Tel. 0721/6091-228