PEARL RUNDSCHAU

Oktober 1980 Band 1 Nr. 2

Tagungsband des Symposiums
"Anwendung von PEARL in der Netzleittechnik"
am 25./26.6.1980 in Deggendorf

INHALT: Seite

Der PEARL-Verein e.V. (PEARL-Association) hat das Ziel, die Verbreitung der Realzeitprogrammiersprache PEARL (Process and Experiment Automation Real-time Language) und ihre Anwendung sowie die Einheitlichkeit von PEARL-Programmiersystemen zu fördern.
Sitz des Vereins ist Düsseldorf. Seine Geschäftsstelle befindet sich im VDI-Haus, Graf-Recke-Straße 84, 4000 Düsseldorf 1.

Vorstand des PEARL-Vereins:

Prof. Dr.-Ing. R. Lauber Institut für Regelungs- technik und Prozeßauto- matisierung Seidenstraße 36 7000 Stuttgart 1	Vorsitz, zuständig für Technik und Normung
Dipl.-Ing. G. Müller Brown Boverie und Cie. AG Leiter der Vertriebsabteilung Netzführungssysteme, SI/NV1 Wallstadter Straße 53-59 6802 Ladenburg	stellv. Vorsitz zuständig für Organisation und Finanzen
Dr.-Ing. P. Elzer DORNIER System GmbH Abt. EEA Postfach 1360 7990 Friedrichshafen	zuständig für Öffentlichkeits- arbeit und Marketing

Die PEARL-Rundschau ist das Mitteilungsblatt des PEARL-Vereins e.V. zur Information seiner Mitglieder. Sie erscheint in zwangloser Folge, mindestens jedoch vierteljährlich. Preis des Einzelheftes für Nichtmitglieder: DM 5.-
Schriftleitung: Dr.-Ing. P. Elzer

Beiträge zur PEARL-Rundschau sollten an den Schriftleiter gesandt werden. Ihre Veröffentlichung wird vereinfacht, wenn sie in kopierbarer Form mit Schreibmaschine geschrieben auf DIN A-4-Papier ohne Aufdruck vorliegen. Vom Schriftleiter kann ein Musterblatt mit dem einzuhaltenden Format angefordert werden.
Es obliegt dem Autor, die Rechte zur Veröffentlichung seines Beitrags in der PEARL-Rundschau sicherzustellen. Der PEARL-Verein erhebt keine Einwände gegen die Kopie einzelner Beiträge bei Angabe der Quelle.

Mit dem Namen des Autors gekennzeichnete Beiträge geben nicht unbedingt die Meinung des Schriftleiters, des PEARL-Vereins oder dessen Vorstand wieder.
Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürfen.

Vorwort der Schriftleitung

Liebes Mitglied,

endlich - werden Sie sagen - ist es fertig, das zweite Heft der PEARL-Rundschau. Nun, wir sagen das auch. Leider hat sich die Urlaubszeit eben doch verzögernder ausgewirkt als ursprünglich angenommen. Da wir Ihnen eine zumindest 'zufriedenstellende' grafische Qualität der Zeitschrift bieten wollen, gibt es eben Rückfragen, Änderungen, es müssen Zeichnungen neu angefertigt werden usw.usw. Nun, wir sind der festen Zuversicht, daß diese technischen Probleme bald gelöst werden können. Immerhin ist der VDI-Verlag an der Herausgabe der PEARL-Rundschau interessiert und es laufen entsprechende Gespräche. Vielleicht wird die nächste Nummer also schon in ganz neuem Gewande erscheinen!

Doch nun zur Deggendorfer-Tagung selbst. Sie war - ohne jede Einschränkung - ein großer Erfolg.
Es hatten sich, trotz des sehr speziellen Themas und einer angekündigten Teilnahmebeschränkung, über achtzig Teilnehmer eingefunden. Da sich viele davon erst in letzter Minute angemeldet hatten, ist es als ein besonders Bravourstück der örtlichen Tagungsleitung zu werten, daß alle untergebracht werden konnten und die Tagung so reibungslos ablief.

Wir möchten deshalb im Namen des PEARL-Vereins der Fa. OBAG für ihre Unterstützung und Herrn Dr. Mayr und seinen Mitarbeitern für ihren unermüdlichen Einsatz noch einmal ganz besonders danken.

Was die technische Leitung angeht, so sprechen die Artikel über die Installation bei der OBAG in Deggendorf für sich selbst.

Manchen Teilnehmern wurde klar, daß hier auf einem volkswirtschaftlich sehr bedeutsamen Einsatzgebiet, der Netzleittechnik, ein technischer Durchbruch erzielt wurde, der zu einem wesentlichen Teil "à conto PEARL" geht. Dazu kommt, daß es sich hier nicht um einen singulären Einsatz handelt, sondern daß in der Bundesrepublik bereits eine ganze Reihe von Netzleitstellen unter Zuhilfenahme von PEARL automatisiert wurden. Wir werden uns bemühen, weiter darüber zu berichten.

Es steht zu hoffen, daß dieser Tagungsband den einschlägigen Fachleuten etwas bei ihrer Entscheidungsfindung hilft und daß das Beispiel der Netzleittechnik sich auch auf andere Anwendungsgebiete im Sinne einer Ermutigung zum Einsatz von PEARL auswirkt.

Mit freundlichen Grüssen

P. Elzer

FUNKTION UND AUFGABEN DER NETZLEITSTELLE DEGGENDORF

Dr.-Ing. U. Mayr
OBAG
8400 Regensburg

Die ENERGIEVERSORGUNG OSTBAYERN AG (OBAG) ist ein regionales Elektrizitätsversorgungsunternehmen. Auf einer Fläche von 22 000 km^2 (Bild 1) versorgt sie ein Drittel von Bayern mit elektrischer Energie.

In Bild 2 sind einige Daten des Unternehmens zusammengestellt. Bezogen auf die Fläche ist die abgegebene elektrische Arbeit des Unternehmens relativ gering, da wenig Ballungsräume, dafür aber sehr flächige Gebiete versorgt werden müssen.

Bezogen auf die Fläche ist die OBAG das zweitgrößte EVU der Bundesrepublik, hinsichtlich des Stromabsatzes liegt das Unternehmen an 14. Stelle. Zur sicheren Führung des ausgedehnten 110- und 20-kV-Netzes und zur raschen Behebung von evtl. Störungen sollen für das Gesamtgebiet der OBAG 4 Netzleitstellen gebaut werden, darüber hinaus wird später zur Koordinierung der vier Netzleitstellen eine übergeordnete Netzleitung errichtet.

Jede dieser 4 Netzleitstellen ist zuständig für das Verteilungsnetz in einem Gebiet von rd. 6 000 km^2 (Bild 3).

In diesem Gebiet werden im Endausbau rd. vierzig Schalthäuser im 20-kV-Netz und dreißig 110/20-kV-Umspannwerke fernüberwacht, ferngemessen und ferngesteuert. Um den enormen Anfall an Informationen in einer Netzleitstelle zu bewältigen, müssen Prozeßrechensysteme eingesetzt werden. Alle Informationen werden auf Farbsichtgeräten dargestellt und angezeigt, Befehle werden über eine funktionelle Tastatur eingegeben. Darüber hinaus erledigt das Prozeßrechensystem zusätzliche Aufgaben, wie Protokollierung aller Betriebsabläufe, Überwachung von Meßwerten, Aufzeichnen von Meßwerten und zusätzliche für den Betrieb wichtige Überwachungs- und Steuerungsvorgänge (Bild 4).

Die diesem Bericht zugrunde liegenden Arbeiten wurden mit Mitteln des Bundesministers für Forschung und Technologie (Förderungskennzeichen P 4.2/36, RE-OBA/1) gefördert. Die Verantwortung für den Inhalt liegt jedoch allein beim Autor.

Trotz all der Technik, die für eine moderne Netzleitstelle nötig ist, wurde jedoch nicht vergessen, daß der Mensch hier nach wie vor das bestimmende Element ist. Es wurde deshalb besonderer Wert darauf gelegt, die Leitstelle zu einem Werkzeug zu machen, das dem Personal gut in der Hand liegt.

In der ersten Netzleitstelle in Schwandorf wurde das gesamte Programmsystem noch in Assembler erstellt.

In der Ausschreibung für die Netzleitstelle Deggendorf wurde jedoch die Erstellung des Programmsystems in Basic-PEARL gefordert. Der Grund dafür liegt darin, daß die Funktionen aller vier Netzleitstellen identisch sind. D.h. aber, daß bei der Erstellung der Software in Assembler, unter Berücksichtigung des vorgegebenen Zeitplanes (Bild 5, das Programmsystem für jede Netzleitstelle praktisch neu geschrieben werden müßte.

Darüber hinaus müssen im Laufe der Zeit in allen vier Netzleitstellen entsprechend einer durch Erfahrungen abgeänderten Aufgabenstellung oder durch zusätzliche Wünsche der Betriebsabteilungen neue Funktionen implementiert werden. Nur im Fall der Anwendung einer höheren Programmiersprache kann dies wirtschaftlich durchgeführt werden. Bei genauerer Betrachtung des Marktes stellte sich 1975 heraus, daß eben PEARL die einzige wirkliche Echtzeitprogrammiersprache ist.

Bei der Anwendung von PEARL in Deggendorf handelt es sich um ein echtes Pilotprojekt, das auch vom Bundesministerium für Forschung und Technologie finanziell unterstützt wurde.

Heute wissen wir, welches Risiko die Entscheidung für PEARL seinerzeit war. Der Erfolg hat uns jedoch Recht gegeben und alle Beteiligten sind sich heute einig, daß die Entscheidung richtig war.

Abschließend möchte ich noch kurz einige Angaben zu der in der Netzleitstelle Deggendorf installierten Technik (Bild 6) machen:

Die Informationen werden vor Ort von Fernwirkgeräten erfaßt und über WT-Kanäle mit TF-, TFH- und Funkanlagen in die Zentrale eingegeben. Hier erfolgt die Eingabe über Mikroprozessor-gesteuerte "Fernwirkzentralen" in das Doppelrechensystem.

In der Warte sind zwei unabhängige Arbeitsplätze mit Fernsprechvermittlungsplatte, Bedienungstastatur und 5 Sichtgeräten eingerichtet.

Derzeit steht die Inbetriebnahme des Softwaresystems kurz vor dem Abschluß. Die zentralen Einrichtungen werden etwa im September 1980 voll in Betrieb sein. Bis Ende des Jahres sollen ca. 60 % aller Fernwirkgeräte angeschaltet werden. Die vollständige Inbetriebnahme des Gesamtsystems mit allen Unterstellen wird Ende 1981 abgeschlossen sein.

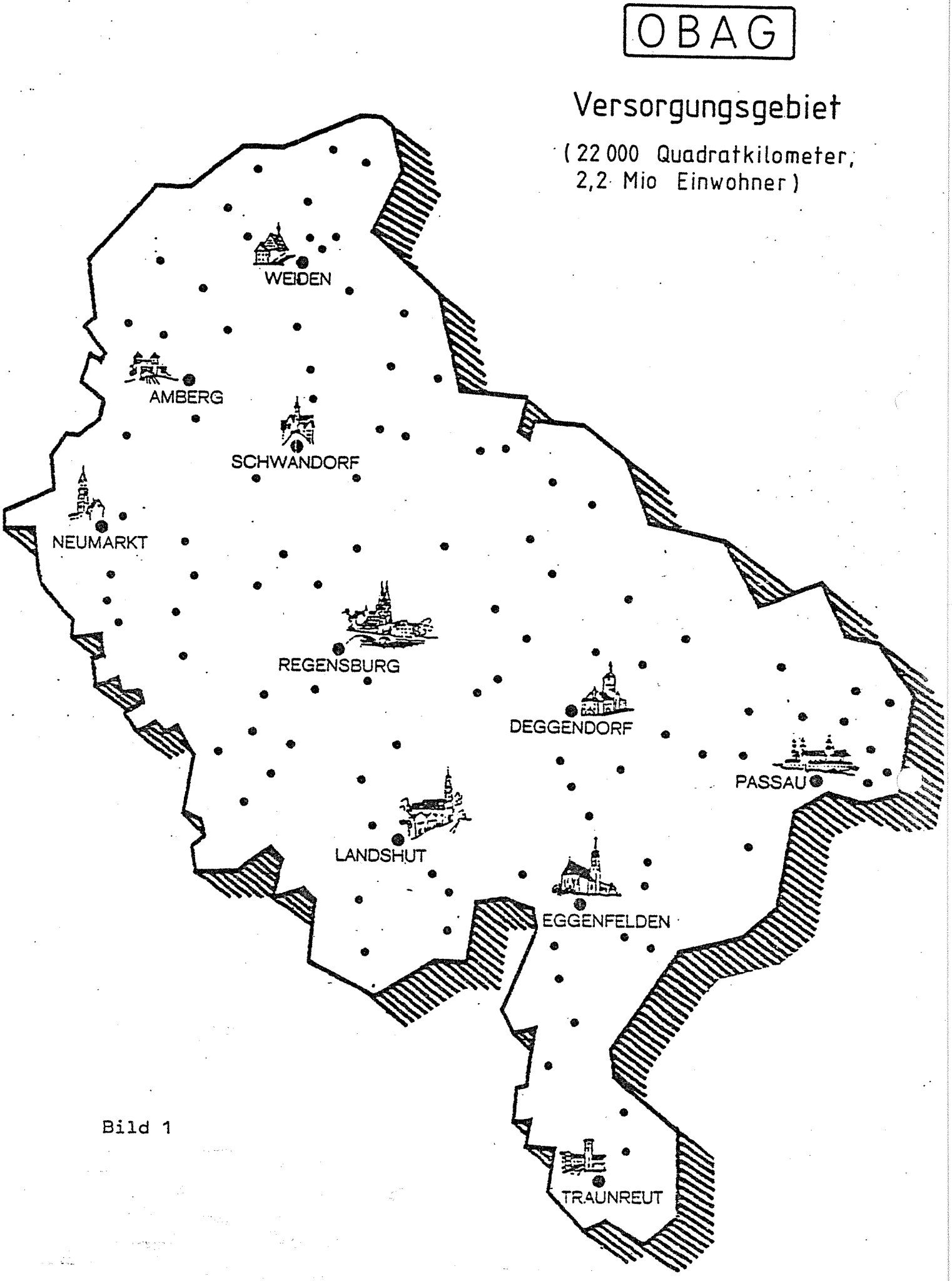

OBAG
Versorgungsgebiet
(22 000 Quadratkilometer,
2,2 Mio Einwohner)
WEIDEN
AMBERG
SCHWANDORF
NEUMARKT
REGENSBURG
DEGGENDORF
PASSAU
LANDSHUT
EGGENFELDEN
TRAUNREUT

Bild 1

ENERGIEVERSORGUNG OSTBAYERN AG

Jahreshöchstleistung	1119.0	MW
Stromabgabe	6433.7	GWh
Tarifkunden (Haushalt, Gewerbe, Landwirtschaft)	494908	
Weiterverteiler	62	
Sonderabnehmer (Industrie)	3668	
eigene Kraftwerke	9	
mit	143.7	MW
Umspannwerke 110/20 kV	60	
Mittelspannungsanlagen 35, 20 kV	220	
110 kV - Leitungen	1181	km
35/20 kV - Leitungen	16779	km
Niederspannungsleitungen	25501	km

Stand: 30.9.1979

Bild 2

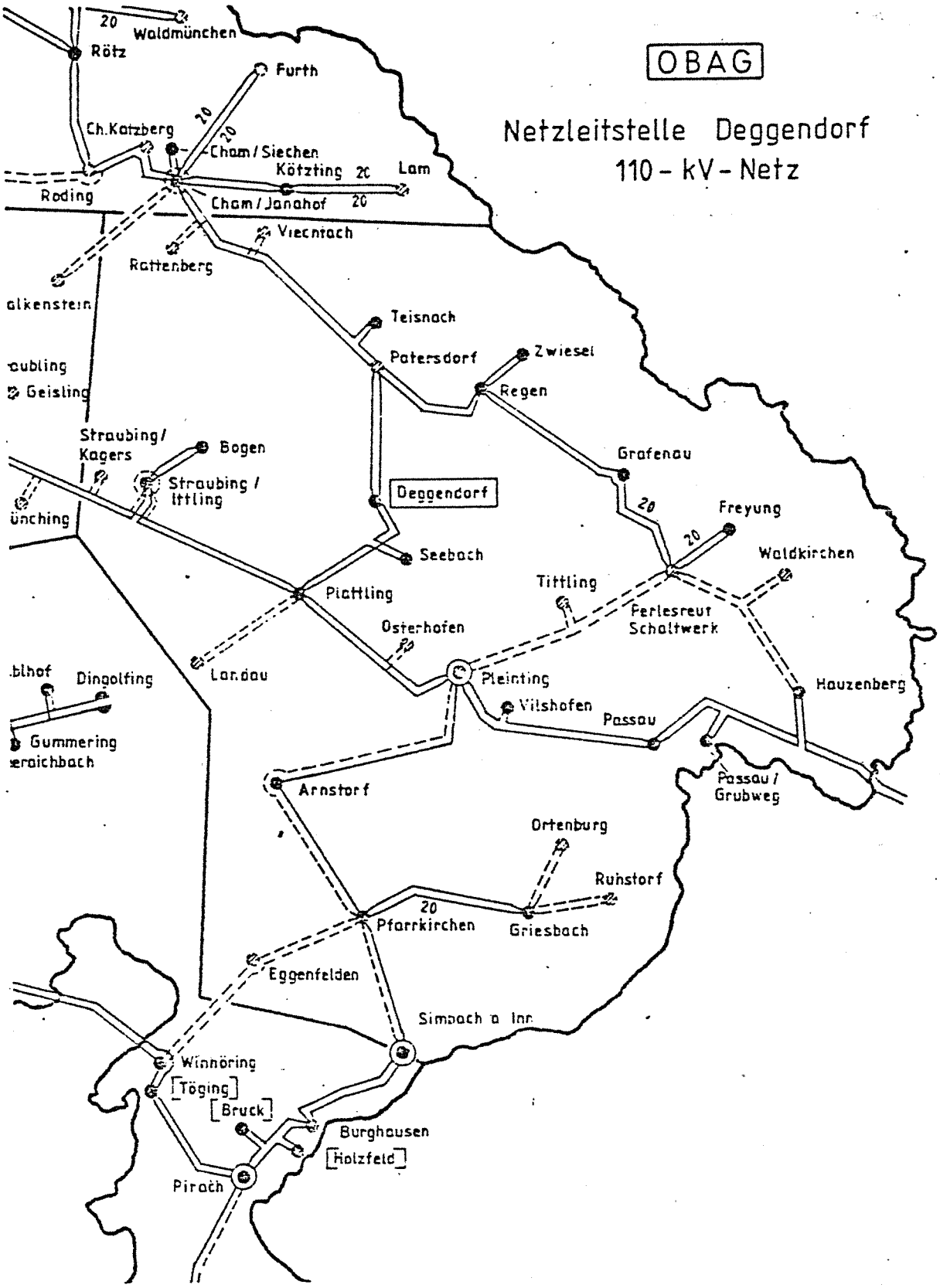

OBAG
Netzleitstelle Deggendorf
110 - kV - Netz
Waldmünchen
Rötz
Furth
Ch.Katzberg
Cham / Siechen
Kötzting
Lam
Roding
Cham / Janahof
Viechtach
Rattenberg
Teisnach
Patersdorf
Zwiesel
Regen
Geisling
Straubing / Kagers
Bogen
Straubing / Ittling
Deggendorf
Grafenau
Freyung
Seebach
Waldkirchen
Plattling
Tittling
Perlesreut Schaltwerk
Osterhofen
Landau
Dingolfing
Pleinting
Vilshofen
Passau
Hauzenberg
Gummering
Arnstorf
Passau / Grubweg
Ortenburg
Ruhstorf
Pfarrkirchen
Griesbach
Eggenfelden
Simbach a. Inn
Winnöring
Töging
Bruck
Burghausen
Holzfeld
Pirach

Bild 3

SOFTWARE SYSTEM

- Erfassung und Decodierung von Fernwirktelegrammen
- Protokollierung von Meldungen / Meßwerten
- Grenzwertüberwachung von Meßwerten
- Meldungsverarbeitung
- Verarbeitung von Tasteneingaben
- Ausgabe / Aktualisierung von Displaybildern
- Konstruktion / Änderung von Displaybildern
- Dialogsystem zur Verwaltung von Prozeßdaten

OBAG Netzleitstelle Deggendorf

Bild 4

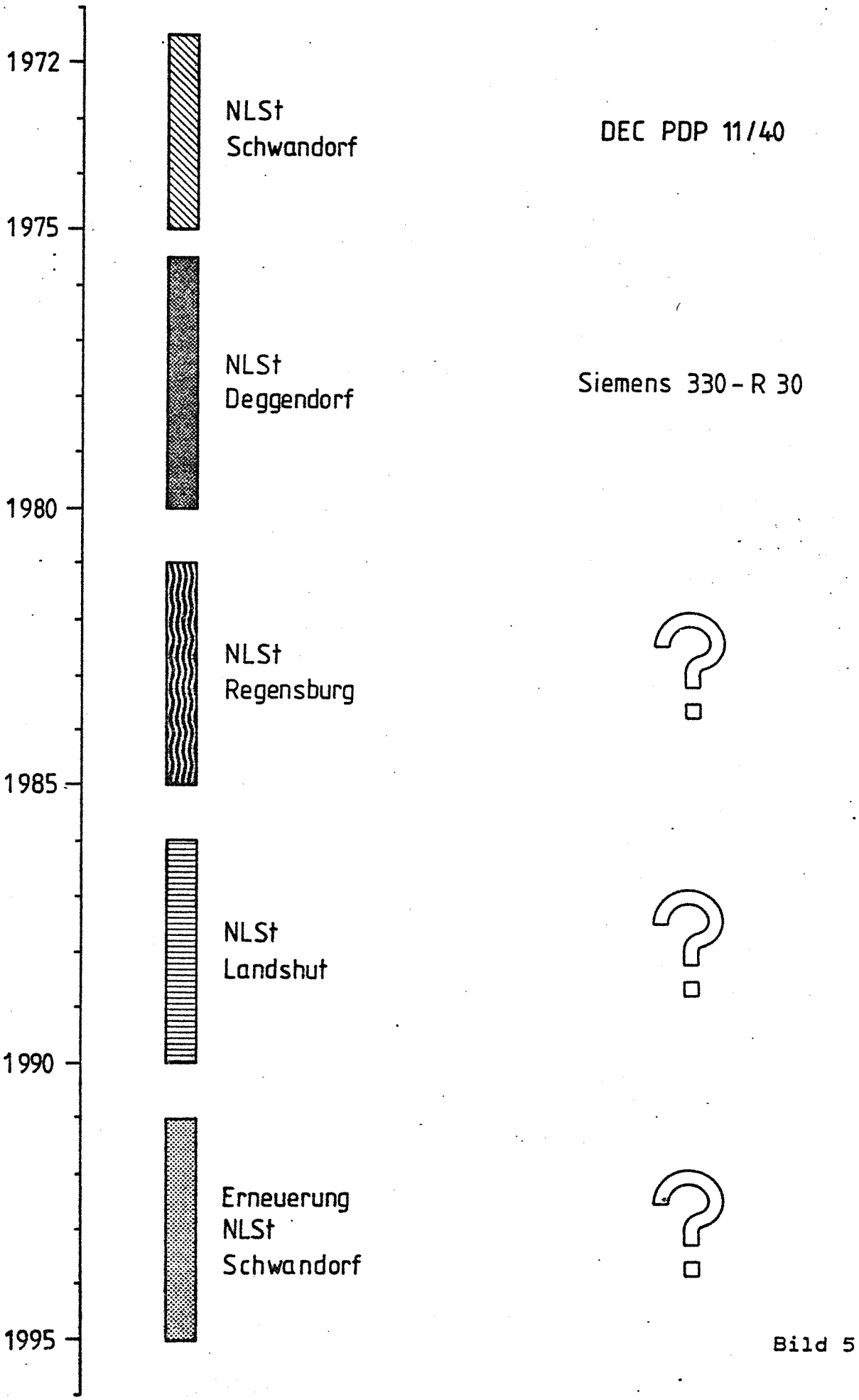

Projekt
Rechensystem
1972
1975
1980
1985
1990
1995
NLSt
Schwandorf
DEC PDP 11/40
NLSt
Deggendorf
Siemens 330 - R 30
NLSt
Regensburg
NLSt
Landshut
Erneuerung
NLSt
Schwandorf

Bild 5

Netzleitstelle Deggendorf

Fernwirksystem:

Typ:	Siemens FW 537
Verkehrsablauf:	vollduplex
Code:	Pulsdauermodulation
Telegrammlänge:	32 bit

Übertragungssystem:

WT:	100 bd
TF:	Z 12 auf Erdseilluftkabel

Zentrale:

Fernwirkgeräte:	SINAUT 8 FW
Rechner:	2 x 330-R30 mit 192 kW
Stromversorgung:	gesichert mit Gleichrichter- Batterie-Wechselrichter 3 x 20 kVA 2 aus 3 - Betrieb

Bild 6

ERFAHRUNGEN MIT BASIC PEARL IM PROJEKT NETZLEITSTELLE DEGGENDORF

Ing.grad. D. Struhalla
Siemens AG - E 14 - 8520 Erlangen

Zusammenfassung:

Der vorliegende Beitrag schildert die Erfahrungen, die mit Basic-PEARL bei der Projektabwicklung für die Netzleitstelle Deggendorf der Energieversorgung Ostbayern AG gemacht wurden. Um die Erfahrungen im rechten Licht erscheinen zu lassen, wird zunächst das Projekt anhand eines historischen Rückblicks, der Anlagenkonfiguration und einiger Projektdaten vorgestellt.

Darüberhinaus enthält der Beitrag einige Anregungen für die Abwicklung von 'PEARL-Projekten'.

Die PEARL-Anwendung hat gezeigt, daß die Sprache eine schnelle, fehlerarme und selbstdokumentierende Programmierung ermöglicht.

PEARL-Programme haben, wie alle in höheren Programmiersprachen geschriebenen Programme, einen gegenüber Assembler größeren Speicherplatzbedarf, zeichnen sich jedoch durch eine hohe Zuverlässigkeit aus.

Die diesem Beitrag zugrunde liegenden Arbeiten wurden mit Mitteln des Bundesministeriums für Forschung und Technologie (Förderungszeichen: P 4.2/36, RE-OBA/1) gefördert.
Die Verantwortung für den Inhalt liegt jedoch allein beim Autor.

1. Projektgeschichte

Siemens erhielt am 6.12.76 den Auftrag zur Ausrüstung einer Netzleitstelle für die Energieversorgung Ostbayern AG (OBAG) in Deggendorf.

Bestandteil des Auftrages war die Forderung nach Implementierung der Anwendersoftware in der Programmiersprache PEARL.

Im folgenden werden einige markante Termine genannt, die unsere Arbeiten bestimmten:

12.76 - 03.78	Analyse, Spezifikation
09.77	Beschreibung Basic PEARL (PDV 120/121)
10.77	Beschreibung Full PEARL (PDV 130)
01.78	Freigabe des ersten Basic PEARL-Compilers von Siemens
02.78	Veröffentlichung der OBAG-Funktionsanalyse (PDV-E 108)
04.78 - 09.79	Implementierung, Test
05.78	Beginn der PEARL-Schulung durch die Schule für Prozeßrechnertechnik
06.78	Festschreibung von Basic PEARL (DIN 66253 Teil 1)
12.78	Liefereinsatz des Basic PEARL-Compilers von Siemens (DIN 66253)
10.79 - 11.80	Inbetriebnahme

2. Anlagenkonfiguration

Ein Rechnersystem bestehend aus

- 1 Zentraleinheit R30 mit 384 k Byte Hauptspeicher
- 2 Sichtgerätearbeitsplätzen in der Warte mit je 4 grafischen Sichtgeräten, Format 80 x 48, für Netz- und Übersichtsbilder und je 1 alphanumerischen Sichtgerät, Format 80 x 24, für Betriebslisten
- 1 Dialogarbeitsplatz im Rechnerraum mit 1 grafischen Sichtgerät, Format 80 x 48, zur Bildkonstruktion und 1 alphanumerischen Sichtgerät, Format 80 x 48, zur Datenpflege.

Aus Sicherheitsgründen ist ein zweites Rechnersystem installiert worden. Das sich daraus ergebende Doppelrechnersystem wird im 'stand-by' Betrieb gefahren.

Jeder Rechner hat seine eigenen Massenspeicher und ein eigenes Prozeßelement. Umschaltbar sind lediglich die Arbeitsplätze mit ihren Sichtgeräten.

Mit dem Doppelrechnersystem ist ein sog. Testbetrieb möglich. Hierbei bekommt jeder Rechner einen Arbeitsplatz zugeschaltet. Während der prozeßführende Rechner mit nur einem Pult den Prozeß weiterführt, können auf dem anderen Rechner mit dem zweiten Pult z. B. neue Daten ausgetestet werden.

3. Projektdaten

3.1 Hardware

Das Netzleitsystem ist für 100 Unterstationen ausgelegt. Der derzeitige Auftragsumfang beläuft sich auf 78 Unterstationen.

Die Informationen aus den Unterstationen werden parallel beiden Rechnern zugeführt.

Das statische Informationsvolumen, bezogen auf den Endausbau, beträgt

20.000 Meldungen
1.500 Meßwerte
1.500 Befehle

Unter den technologischen Begriffen Meldung, Meßwert und Befehl werden hier binäre Signale zwischen 1 Bit und 8 Bit Länge verstanden.

Das dynamische Informationsvolumen beträgt bei 100 Bd. Übertragungsgeschwindigkeit und 48 Zentralgeräten

$$\text{permanent } 60 \quad \frac{\text{Telegramme}}{\text{s}}$$

oder anders ausgedrückt, die Telegrammfolgefrequenz beträgt

16 ms

Die Telegrammbreite ist 48 Bit und enthält entweder 4 Meßwerte oder 32 Meldungen.

3.2 Software

Auf dem standardmäßigen Betriebssystem ist ein Prozeß- und ein online laufendes Dialogsystem aufgesetzt.

Das Prozeßsystem enthält die Aufgaben Steuern, Melden, Überwachen, Protokollieren und Bedienen.

Das Dialogsystem beinhaltet die Aufgaben Datenpflege und Bildkonstruktion.

Beide Dialoge, sowohl Datenpflege als auch Bildkonstruktion, sind losgelöst von internen Softwarestrukturen. Dies ist die Voraussetzung dafür, daß der Anlagenbetreiber selbst die Rechner mit seinen Netzdaten 'füttern' kann.

Der Aufwand für das Prozeß- und Dialogsystem beläuft sich auf 170 Mannmonate oder in Programmen gesprochen auf 65 PEARL-Objekte mit 1000 K Byte Länge und 4 Assembler-Objekten mit 44 K Byte Länge.

Der Assembler wurde für die besonders zeitkritische und platzoptimale Fernwirktelegrammbearbeitung eingesetzt.

4. Ergebnisse

4.1 Sprache

Erlernbarkeit

Die Sprache ist leicht erlernbar.

Zum wirklichen Arbeiten mit dem Werkzeug PEARL benötigt man allerdings auch das Verständnis für das zugrundeliegende Sprachkonzept. Als Beispiel sei hier die Typverträglichkeit (Konvertierungsroutinen) genannt.

Sprachumfang

Der Sprachumfang von Basic PEARL hat sich bis auf einige Ausnahmen als ausreichend erwiesen.

Das Siemens PEARL, im folgenden PEARL 300 genannt, ist um einige Sprachelemente im Rahmen von Full PEARL wie z.B. Characterselektion, Gruppenselektion von Charactern und Bits, Initialisierung von Feldern und Prozeduren für Datum und Uhrzeit erweitert.

Die Festlegung des Gültigkeitsbereiches eines Selektornamens in Basic PEARL, der sich nicht auf die zugehörige Struktur

sondern auf den übergeordneten Block bezieht, sehen wir als Nachteil an. Dadurch geht ein wesentlicher Teil der in der Strukturdefinition liegenden Transparenz verloren.

Die Formulierung von Datennahtstellen zwischen einzelnen Programmen - Datei und Common Data - ist zu elementar. Der versierte Anwender vermißt die direkte Datenübergabe zwischen Programmen.

Die Koordinierung des Softwaresystem, Netzleitstelle Deggendorf, geschieht fast ausschließlich über Semaphorvariablen.

Damit PEARL-Systeme noch effektiver arbeiten, müßte die Semaphorvariable dem ACTIVATE-Statement gleichgestellt werden - will sagen, die Semaphorvariable muß um die Zeitmodifikation erweitert werden.

Notation

Die Sprachnotation von PEARL ist übersichtlich, klar, leicht verständlich und selbstdokumentierend.

Assembleranschluß

PEARL 300 arbeitet mit einem variablen blocktiefenabhängigen Keller. Über diesen Keller ist PEARL 300 auf Prozedurebene mit beliebigen Sprachen mischbar.

Zuverlässigkeit

Hier liegt die Stärke der Sprache.

Implementierung und Test haben gezeigt, daß die Fehlerquote bei PEARL-Programmen sehr niedrig ist. Hier erweist sich die bereits zur Compilierzeit erfolgende Datentyp-Prüfung (Konvertierungsroutine) als sehr nützlich.

Die Sprachelemente von PEARL liegen fast ausnahmslos auf höherer Ebene. Die Sprache eignet sich sehr gut als Bindeglied zwischen Systemmann und Technologie. Sie unterstützt guten Programmstil, Modularität und strukturierte Programmierung.

Programmiersprachen sind weder die Ursache noch die Lösung von Softwareproblemen. Wegen ihrer zentralen Rolle bei allen Software-Entwicklungen wirken sie allerdings richtungsweisend.

PEARL bewirkt bei der Lösung von Echtzeitproblemen eine Kosteneinsparung durch

- reduzierte Codier-, Test- und Fehlerbehebungskosten
- geringere Projektführungs- und Wartungskosten aufgrund erhöhter Lesbarkeit und Durchschaubarkeit der Programme

Die Kosteneinsparung beläuft sich gegenüber einer Assemblerprogrammierung zwischen 20 und 40%.

4.2 Compilieren/Binden

PEARL 300 - Module müssen nach der Compilierung mit dem Siemens PEARL-Compiler PC3Ø mit Hilfe des Binders zu einem ladbaren und danach ablauffähigem Programm gebunden werden.

Die Zeit zum Compilieren und Binden liegt im Schnitt bei ca. 5 Min. In Verbindung mit einem leistungsfähigen Drucker ergeben sich somit gute Compilier- und Bindezeiten.

Der Comiler PC3Ø liefert ein gut dokumentiertes Fehlerprotokoll, das eine schnelle und sichere Fehleridentifikation ermöglicht.

4.3 Programmcode

Die mittlere Programmlänge im Projekt Netzleitstelle Deggendorf liegt bei 18 K Byte, wobei das Spektrum von 4 K Byte bis 44 K Byte reicht. Eine subjektive Beurteilung der Länge von PEARL-Programmen und funktionsähnlichen Assemblerprogrammen ergibt einen Faktor um 2. Dieser Faktor gilt allerdings nur in Verbindung mit einem im Zentralspeicher reentrant bereitgestellten Laufzeitsystem.

4.4 Zeitverhalten

Die Sprache PEARL bietet viele Möglichkeiten der Strukturierung und Dimensionierung von Daten. Beides geschieht auf Sprachebene und ist somit rechnerunabhängig.

Im Laufe der Arbeiten haben sich allerdings folgende Erkenntnisse herauskristallisiert.

- Der Aufbau eines umfangreichen Software-Systems ist auch in PEARL ohne Hardware- und Betriebssystem-Kenntnisse nicht möglich.
- Wenn harte Zeitbedingungen eingehalten werden sollen - und wann ist das nicht der Fall - dann müssen rechnerinterne Strukturen mit berücksichtigt werden.
- Durch geschickte Anwendung der Sprache kann unter Berücksichtigung der vom Compiler abgesetzten Befehle ein Programm ganz entscheidend zeitoptimiert werden.

  Beispiel: Eine falsch placierte Wertzuweisung von Strukturselektoren in einer Programmschleife bedeutet bei Meßwertverarbeitungsprogrammen, daß diese Wertzuweisung u.U. 1500-mal unnütz durchlaufen wird.

4.5 Test

In PEARL 300 ist ein 'online Test' zur Laufzeit möglich. Es sind die Funktionen Zeilentrace, Haltepunkte und Haltepunkte mit Bedingung realisiert.

Das Anschauen und Verändern von Variablen ist in Verbindung mit der standardmäßigen Systemtesthilfe möglich.

5. Anregungen

5.1 Konzept

Die in der Konzeptphase angedachten Datenstrukturen sollten im Hinblick auf eine transparente Bearbeitung folgenden Randbedingungen genügen

- Trennung zwischen Buchführung und Daten (keine Adressverweise in Datensätzen)
- alle Datenstrukturen so vereinbaren, daß eine wiederholbare Datensatzstruktur entsteht.

5.2 Spezifikation

Bereits in der Spezifikationsphase müssen einige übergeordnete Maßnahmen getroffen werden

- Zentrale Erfassung der globalen Daten für das gesamte System (Vereinbarung von Namen und Datentypen)
- Zentrale Erfassung der im System vorkommenden Dateien und Geräte. (Festlegung von Zugriffsrichtung, Länge, Lage und Datentyp gem. der Dateideklaration)
- Anwendung der strukturierten Programmierung (Nassi Shneiderman)
- Bei Programmnahtstellen gleiche Namen und Datentypen für Daten mit gleicher Bedeutung wählen.
- Für die zentralen Nahtstellen des Systems einen online lauffähigen Nahtstellentrace vorsehen
- Die funktionelle Gliederung der Tasks so wählen, daß ggf. einzelne Funktionen mit möglichst wenig Aufwand in eigenständigen Tasks ausgelagert werden können

5.3 Implementierung

Für die Implementierung gelten folgende Empfehlungen

- Zentrale Erstellung des SYSTEM-Teils
- Zentrale Erstellung eines Definitionssatzes der globalen Daten
- Zentrale Erstellung eines Spezifikationssatzes für die globalen Daten und Prozeduren
- Trotz Formatfreiheit den Protokollaufbau so wählen, daß die Blockstrukur erkennbar ist
- Bei Deklaration pro Variable eine neue Zeile beginnen und einzeln kommentieren.

5.4 Test

Um jederzeit einen bequemen online Test durchführen zu können, sollten taskspezifische Tracefunktionen mit einprogrammiert sein.

Bei Einschalten dieser Fracefunktionen sollte es möglich sein, Programmabläufe nicht nur im Dialog mit dem Rechner verfolgen zu können. Anhand dieses Traceprotokolls muß eine 'post-mortem-Analyse' möglich sein.
(Nur zu empfehlen, wenn die Formatschlüssel reentrant im Zentralspeicher stehen).

5.5 Dokumentation

Bei Einhaltung gewisser Notationsregeln stellt das Quellsprachenlisting die Basisdokumentation dar. Sie muß lediglich um eine übergeordnete Funktions- und Datenbeschreibung ergänzt werden.

6. Resümee

Die PEARL-Anwendung hat gezeigt, daß die Sprache eine schnelle fehlerarme und slebstdokumentierende Programmierung ermöglicht.

PEARL-Programme zeichnen sich durch eine hohe Zuverlässigkeit aus - allerdings zu Lasten eines erhöhten Speicherplatzbedarfs.

DIE PEARL-IMPLEMENTATION VON AEG-TELEFUNKEN FÜR UND AUF AEG 80-20

Dipl.Ing. S. Eichentopf
ATM Computer GmbH 7750 Konstanz

1. Einleitung

AEG-TELEFUNKEN hat von Anfang an - also seit 1969 - bei der Erarbeitung der Sprachdefinition von PEARL mitgewirkt.
Als sich die Sprachdefinition in der Mitte der 70er Jahre nach einer größeren Überarbeitung zu stabilisieren schien, hat AEG-TELEFUNKEN auf der Basis des bis dahin Erarbeiteten mit vermehrter Kapazität die Implementierung der neuen Sprache in Angriff genommen. Dabei wurden nachträgliche Änderungen, die im PEARL-Arbeitskreis einvernehmlich an PEARL vorgenommen wurden, jeweils möglichst umgehend in die Implementierung einbezogen, was durch das gewählte halbautomatische Implementierungs-Verfahren begünstigt wurde. So kann heute eine PEARL-Implementation angeboten werden, deren Sprachumfang auf dem neuesten Stand der offiziellen Full-PEARL-Sprachdefinition ist und der weit über Basic PEARL hinausgeht.

2. Sprachumfang

Bei der Auswahl des Sprachumfangs stand der Nutzen aus Benutzersicht im Vordergrund. Es wurde nur auf die Sprachelemente von Full PEARL verzichtet, deren Implementierung mit unverhältnismäßig hohem Aufwand möglich gewesen wäre oder die nicht auf ausreichend effiziente Weise mit dem vorhandenen Betriebssystem hätten realisiert werden können. Aufgrund dieser Randbedingungen wurde sehr frühzeitig ein zu implementierender Sprachumfang festgelegt, der gegenüber Full PEARL nur wenig eingeschränkt ist.
Die wesentliche Einschränkung ist wohl der Verzicht auf Subtasks, d.h., daß die Einführung von Tasks nur durch entsprechende Deklarationen "auf Modulebene" zugelassen ist, - eine Einschränkung, mit der der Benutzer jedoch gut leben kann. Eine genauere Abgrenzung des AEG 80-20-PEARL-Sprachumfangs gegenüber Basic PEARL und Full PEARL findet sich im Anhang der AEG-80-20-PEARL-Sprachbeschreibung /1/.

Außerhalb von Basic PEARL gibt es eine Reihe von Sprachelementen, die nicht nur den Programmierkomfort erhöhen und zur besseren Lesbarkeit und Selbstdokumentation der Programme beitragen, sondern die überdies die Effizienz der Programme wesentlich erhöhen können. Hierzu einige Beispiele:

- Speicheroptimierung durch Arrays,deren Indexgrenzen dynamisch auf Eingabewerte zugeschnitten sind, bei minimaler Laufzeiterhöhung
- Verminderung der Zahl erforderlicher Tasks durch Einplanungs-Listen bei Aktivierungsanweisungen zusammen mit der Standardfunktion ORIGIN zur Abfrage des auslösenden Einplanungselements
- Zuweisung von Strukturen und Arrays als Ganze statt nur elementweise:

```
DCL (AR1, AR2) (50)FIXED;
DCL (ST1, ST2) STRUCT [E1 TYP1, E2 TYP2, E3 TYP3];

/* FULL PEARL */                    /* BASIC PEARL */

AR1 := AR2;                         FOR I TO 50
                                    REPEAT  AR1 (I) := AR2(I);
                                    END;

ST1 := ST2;                         ST1.E1 := ST2.E1;
                                    ST1.E2 := ST2.E2;
                                    ST1.E3 := ST2.E3;
```

- Vermeidung von Indextransformationen durch beliebige untere und auch negative obere Arrayindexgrenzen,
- Verminderung der Zahl von Adreßberechnungen mit Hilfe von Identitätsspezifikationen (SPC ... IDENT...), z.B. bei Arrays von Strukturen:

```
TYPE STR STRUCT [E1 TYP1, E2 TYP2, E3 TYP3];
DCL A (12,10)STR;
DCL (I, J) FIXED;

/* FULL PEARL*/                     /* BASIC PEARL */
```

```
BEGIN
SPC AIJ STR IDENT(A(I, J));
AIJ.E1 := ...                    A(I,J).E1 := ...
... AIJ.E2 ...                   ... A(I,J).E2 ...
AIJ.E3 := ...                    A(I,J).E3 := ...
END;
```

3. Compiler

Der Compiler läuft auf AEG 80-20/4 und AEG 80-20/5 mit mindestens 96 KBytes, besser 128 KBytes Hauptspeicherausbau und Hintergrundspeicher mit wahlfreiem Zugriff (Kassettenplatte, Wechselplatte, Festkopfplatte).

Die Läufe des Compilers und die Compilertabellen belegen auf Hintergrundspeicher statisch ca. 350 KBytes. Dynamisch beim Compilieren werden zusätzlich, abhängig von den zu übersetzenden Programmen, zwischen 300 KBytes und 500 KBytes Hintergrundspeicher benötigt.

Der Compiler ist selbst außer ein paar Ein-/Ausgabe-Routinen in PEARL geschrieben. Er wurde einmal mit einem sogenannten bootstrap-Verfahren auf den Rechner gebracht und kann sich dort nun selbst übersetzen.

Der Compiler wird mit einem einfachen Kommando gestartet. Die zu übersetzenden Quellprogramm-Moduln können in Quelldateien auf Hintergrundspeicher vorgegeben werden, wo sie auch mit einem Texteditor bearbeitet werden können, oder sie können über Papierperipheriegeräte eingegeben werden. Die größten an einem Stück übersetzbaren Quellmoduln sind je nach Quellzeilenstruktur und anderen Programmeigenschaften 2000 bis 3000 Quellzeilen lang.

Der aus mehreren Läufen bestehende Compiler-"Oberteil" erzeugt als Zwischenprodukt eine Reihe von Listen sowie ein Abbild des Quellprogramms in umgekehrter Polnischer Nation, in der Arrayindizierungen, Strukturelementselektionen, Prozeduraufrufe usw. wie spezielle Operationen behandelt werden. Diese Schnittstelle zwischen Compiler-Oberteil und Codegenerator sowie die Formulierung des Compilers in PEARL machen den Compiler weitgehend rechnerunabhängig und damit übertragbar.

Bei den Läufen des Compileroberteils werden zur Analyse der Eingabeprodukte Bottom-up-Parser, genauer Bounded-Context-Parser verwendet. Wesentliche Bestandteile dieser Parser sind Parsertabellen, an Hand deren analysiert wird und die mit Hilfe von Parsergenerator-Programmen aus vorgegebenen (kontextfreien) Grammatiken automatisch hergestellt wurden.

Der Codegenerator des Compilers erzeugt direkt Bindemoduln - nicht Assembler -, die auf Hintergrundspeicher abgelegt werden oder über Papierperipherie ausgegeben werden können.

Der erzeugte Objektprogrammcode kann durch Parameter im Compilerstartkommando z.T. beeinflußt werden (mit/ohne Indexgrenzenprüfung, Prozeduren generell reentrant oder nicht).

Die Zeit für die Übersetzung von 1000 Quellzeilen eines übersichtlich geschriebenen, durchschnittlichen Programms (im wesentlichen eine elementare Anweisung pro Zeile) aus einer Hintergrund-Quelldatei in Bindemoduldateien auf Hintergrundspeicher liegt - ohne Protokollierungszeiten - bei AEG 80-20/4 unter 5 Minuten.

Der Compiler erzeugt beim Übersetzen ein Protokoll, dessen Umfang durch Parameter im Compilerstartkommando gesteuert werden kann. Zum maximalen Protokollumfang gehören folgende Teile:

- formatgetreuen Quell-Listing mit Zeilennummern
- Listing aller PEARL-Datenobjekte im Programm mit Angabe der Definitionsstelle und sämtlicher Aufrufstellen im Quellprogramm (Zeile, Spalte) sowie der Lage im Speicher des erzeugten Objektprogramms
- Code-Listing in assemblernaher Form mit Rückverweisen auf entsprechende Quellzeilen und -spalten
- Verzeichnis der erzeugten Bindemoduln mit statischen Längen
- Füllungsgrad bzw. Längen wichtiger Compilerlisten

Dazu werden ggf. genaue Fehlermeldungen gegeben, die - von wenigen gravierenden Fällen abgesehen - nicht zum Abbruch des Übersetzungsvorgangs führen, so daß mit einer einzigen Übersetzung eine ganze Reihe unabhängiger Fehler im Quellprogramm festgestellt werden kann.

4. Objektprogramme

Die Sprachelemente von PEARL, die nicht einfach durch inline-Maschinenbefehlsfolgen und nicht direkt durch entsprechende Betriebssystemdienste realisiert werden können, werden mit Hilfe der Moduln eines Laufzeitpakets realisiert, die die erforderliche Leistung entweder vollständig erbringen oder ggf. eine Abbildung auf geeignete Betriebssystemdienste vornehmen. Insbesondere werden auf diese Weise die Tasks des PEARL-Programms 1:1 auf vom Realzeit-Betriebssystem MARTOS-K verwaltete "Programme" bzw. Aktivitäten der Tasks auf Prozesse über diesen Programmen abgebildet.

Das Laufzeitpaket ist stark modularisiert. Beim Laden und Binden eines speziellen PEARL-Objektprogramms werden nur die hierfür erforderlichen Moduln des Laufzeitpakets hinzugeladen und gebunden, und zwar automatisch aus entsprechenden Bibliotheken.

Die vom Compiler erzeugten Bindemoduln des Objektprogramms können mit dem normalen im Dialog zu bedienenden Binder-Lader des AEG 80-20-Programmiersystems geladen und gleichzeitig gebunden werden, wobei bezüglich der Ablage im Speicher einige wenige Konventionen zu beachten sind. Bequemer ist das Laden und Binden mit Hilfe einer Kommandofolge, die auf Lochkarten oder in einer Datei vorgegeben wird und die sich an die sogenannte automatische Programmsystemgenerierung wendet, mit der auch Betriebssysteme generiert werden. Es können auch Moduln hinzugeladen werden, die nicht aus PEARL-Quellen entstanden sind und die vom PEARL-Programm aus als globale Prozeduren aufgerufen werden können.

5. Weiterentwicklung

In Arbeit ist eine dritte Möglichkeit für das Laden und Binden:
Der erforderliche Dialog mit dem Lader-Binder des Programmiersystems wird von einem Programm geführt, das gewisse vom Compiler über die übersetzten Programme angelegte Beschreibungsdateien direkt verwertet und die für PEARL-Programme geltenden Lade-Binde-Konventionen ausnutzt. Dadurch wird das Laden von PEARL-Programmen in vielen Fällen so einfach wie das Starten des Compilers.

Außerdem werden die quellbezogenen Testhilfen ausgebaut, wofür bereits die erforderlichen Voraussetzungen im Compiler und z.T. im Laufzeitpaket geschaffen sind.

Daneben werden weitere Objektprogramm-Optiemierungen in den Compiler eingebaut.

6. Anwendungen

Die AEG 80-20-PEARL-Implementation wurde inzwischen mehrfach ausgeliefert und hat nach dem Urteil der Benutzer einen stabilen Stand erreicht.

Sie wird seit Anfang 1978 auch in der Schulungsabteilung für PEARL-Kurse mit Übungen eingesetzt.

Eine Anwendung der Implementation ist der in PEARL geschriebene Compiler selbst. Hier wird nicht nur der gute und leistungsfähige "algorithmische Sprachkern" von PEARL in Anspruch genommen, sondern auch das Tasking. So laufen die besonders "EA-intensive" lexikalische Analyse und der erste Syntaxanalyselauf nicht nacheinander sondern als Tasks zeitlich parallel mit entsprechender Synchronisierung. Auch die verschiedenen Ein-/Ausgabe-Vorgänge des Compilers sind als Tasks organisiert, die zeitlich parallel zu Verarbeitsungs-Tasks laufen, jeweils geeignet synchronisiert.

Erstmals mit der damals verfügbaren Entwicklungsversion wurde von Mitarbeitern von Prof. Jünemann (Universität Dortmund) für die INTERKAMA 77 das Modell eines flexiblen Fertigungssystems, gesteuert mit PEARL-Programmen, implementiert. Es handelt sich zwar nur um ein Modell, aber dennoch um ein nicht triviales (z.B. je 80 Digital-Ein- und -Ausgaben, 40 Tasks). (Ein Aufsatz über dieses Modell-Projekt ist u.a. in /2/ sowie in PDV-E112 und KfK-PDV 171 enthalten).

Die Durchführung des Projektes in der Kürze der verfügbaren Zeit wäre ohne Verwendung von PEARL (oder einer mindestens gleichwertigen höheren Realzeit-Programmiersprache) nicht zu schaffen gewesen.

Bei diesem Projekt wie auch bei anderen Anwendungen erwies sich der über Basic PEARL hinausgehende Sprachumfang als hilfreich.

Literaturhinweise

/1/ AEG 80-20 PEARL Sprachbeschreibung
AEG-TELEFUNKEN Konstanz 1979
(Diese Beschreibung ist in erster Linie als Handbuch zur AEG 80-20-PEARL-Implementation gedacht und orientiert sich deshalb in ihrer Gliederung mehr an der Systematik der Sprache PEARL als an didaktischen Gesichtspunkten).

/2/ Zeitschrift "Datenverarbeitung AEG-TELEFUNKEN"
Nr. 2/3, 1977

DAS BBC-PEARL-PROGRAMMIERSYSTEM UND SEINE ANWENDUNGEN IN DER NETZLEITTECHNIK

Dipl.-Math. G. Koch
BBC Abt. SI/NP2 6802 Ladenburg

1. Das BBC-PEARL-Programmiersystem

Seit 1966 beschäftigt sich die Fa. BBC mit dem Einsatz von prozedurorientierten Realzeit-Programmiersprachen zur Rationalisierung der Programmentwicklung für Automatisierungsprojekte (s.Tab. 1). Nach den ersten positiven Erfahrungen mit dem von BBC entwickelten PAS1-Programmiersystem unter Verwendung eines auf Prozeßrechneranforderungen erweiterten PL/I-Subset engagierte sich BBC von Anfang an bei der Entwicklung von PEARL. Die bereits im Jahre 1973 nach Fertigstellung der Definition von PEARL 73 in Angriff genommene Pilotimplementierung eines PEARL-Subsets für die PDP11-Rechnerfamilie wurde durch das PDV-Projekt der Bundesregierung im Rahmen des 2. DV-Förderungsprogramms unterstützt. Diese früheste PEARL-Implementierung besaß folgende Merkmale:

- kleiner PEARL-Subset
- kleines aber reaktionsschnelles PEARL-Betriebssystem
- PEARL-Compiler mit wirkungsvollen Möglichkeiten der Übersetzungs-Beeinflussung (Compile-time-statements)
- Test- und Bedienungshilfsmittel

Erste Einsatzerfahrungen bei der Abwicklung von industriellen Automatisierungs-Projekten bestätigten die Erwartungen:

- Erhebliche Rationalisierung von Programmentwurf und Programmtest gegenüber Assembler-Programmierung
- Unterstützung der strukturierten Programmierung

Nach der Durchführung mehrerer Projekte kamen von Seiten der Anwender folgende zusätzliche Forderungen:

Die diesem Bericht zugrunde liegenden Arbeiten wurden mit Mitteln des Bundesministers für Forschung und Technologie im Rahmen des 1. und 2. DV-Förderungsprogramms gefördert. Die Verantwortung für den Inhalt liegt jedoch allein beim Autor.

- Vergrößerung des Subsets
- Erweiterung des Betriebssystems unter Beibehaltung der kurzen Reaktionszeiten
- Bereitstellung von weiteren Hilfsmitteln für größere Projekte

Durch Auswertung des Erfahrungsrückflusses aus über 100 Anlagen in verschiedenen Branchen (s.Bild 1) entwickelte sich das zum heutigen Zeitpunkt bei BBC im industriellen Einsatz befindliche PEARL-Programmiersystem:

- Verfügbar für die PDP11-Rechnerfamilie von der LSI/11 bis zur PDP 11/70
- abgerundeter PEARL-Subset, der deutlich über BASIS-PEARL hinausgeht
- ausgereiftes, bedienungsfreundliches PEARL-Programmiersystem
- auch als core-only Version verfügbar
- Wartung und Pflege durch BBC

In der Zukunft sieht BBC die Bereitstellung und Pflege von effizienten PEARL-Programmiersystemen als Aufgabe der Rechnerlieferanten. In diesem Sinne wurde die Fa. Digital Equipment durch BBC angeregt, eine PEARL-Implementierung bereitzustellen.

In Bild 2 sind die wesentlichsten Komponenten des BBC-PEARL-Programmiersystems dargestellt. Alle Teile sind so optimal aufeinander abgestimmt, daß sämtliche Fehlermeldungen sowie Bedien- und Testhandlungen sich unmittelbar auf die PEARL-Quellenprogramme beziehen.

Folgende Leitlinien lagen dem Entwurf des Programmvorbereitungskomplexes zugrunde:

- Erkennung aller Formalfehler in einem PEARL-Programmpaket
- Bereitstellung von Listen und Statistiken zur Unterstützung bei Fehlersuche und bei Programmänderungen

- Vollständige Referenzen zwischen Quellenprogramm und Speicherplatz (Befehle oder Daten)

Dabei kommt dem Compiler- und Prelinker-Listings zentrale Bedeutung zu.

Das Compiler-Listing eines PEARL-Moduls besteht aus folgenden Teilen:

Quellenprogramm mit:
- Blockstrukturtiefe
- Statementnr. und Ablage-Adresse

Fehlermeldungen:
- Statementnr.
- Fehlerstart
- Namen der fehlerbehafteten Variablen

Crossreferenz:
- alle Vereinbarungen alphabetisch sortiert
- Adresse für Ablage
- Attribute der Größen
- Benutzungsliste

Statistische Daten

Beim Prelinken eines PEARL-Anwenderpakets entstehen folgende Listen:

Liste der Module:
- Module-Name
- Compilierdatum
- Fehlerzustand

Liste der Task's und Prozeduren:
- Name
- Module-Name
- Speicherbedarf

Module-Statistik:
- Module-Name
- Speicherbedarf
- Zuordnungsdaten

Task-Statistik
- Taskname
- Priorität
- Speicherbedarf
- Module-Name

Fehler-Diagnostik
- Name der globalen Variablen
- Referenz in Module
- Deklariert in Module
- Fehlerart

Global-Cross-Reference
- Name der globalen Variablen
- Modul, in welchem die Variable deklariert ist
- vollständige Attributliste
- Liste der Module, die Spezifikationen enthalten

Die Komponenten des Laufzeitsystems sind nach gleichen Grundsätzen entwickelt worden, d.h. das Quellenprogramm ist die Bezugsbasis:

- für Test- und Bedienhandlungen
- für Fehlermeldungen des Betriebssystems

2. Beispiel für den Einsatz von PEARL bei EVU-Projekten: Zentrale Netzleitstelle für die Stadtwerke Mannheim (SMA)

Auch für Projekte aus dem EVU-Bereich wurde seit 1974/75 das BBC-PEARL-Programmiersystem eingesetzt. Das Projekt der zentralen Netzleitstelle für die Stadtwerke Mannheim wird als Beispiel aus 74 Anlagen herausgegriffen, da neben den üblichen EVU-Aufgaben auch die des überlagerten Netzschutzes mit einer Reaktionszeit unter 300 msec zu erfüllen sind. Dieser Reserveschutz greift bei Versagen des Abzweigschutzes ein, um die Auslösung der Sammelschieneneinspeisung zu verhindern.

Die Projektaufgabe besteht in der Realisierung der zentralen Netzführung für das E-Netz des Stadtgebietes von Mannheim mit folgenden Zielen:

- Erhöhung der Verfügbarkeit und Betriebssicherheit des Netzes
- Erhöhung der Wirtschaftlichkeit durch Nutzung von Netzreserven
- Entlastung des Betriebspersonals von Routineaufgaben

Dafür wurden in der zentralen Netzleitwarte folgende Aufgaben realisiert:

- Darstellung des aktuellen Netzzustandes in Form von Übersichts-, Teil- und Anlagenbildern auf Farbmonitoren:
  - graphische Darstellung des Netzes
  - Zustand der Objekte durch Farbgebung
  - aktuelle Daten (Spannung, Strom, Leistung, Temperatur)
  - Markierung von irregulären Zuständen (Farb-
  - hinterlegung, Blinken)

- Auslösung von Schalthandlungen durch Identifikation des Objektes im Bild und Kommandogabe über Funktionstastatur

- signifikante Meldung von Prozeßereignissen
  - akustisches Signal (durch Gong)
  - Anzeige der betroffenen Stationen im Netzübersichtsbild
  - automatische Bildaufschaltung mit blinkenden Objekten
  - Warn- u. Störmeldungen auf a/n-Monitor und Protokolldrucker

- automatische Reaktion auf Prozeßereignisse
  - überlagerter Netzschutz bei Versagen des Primärschutzes

- Handnachführung von Netzzustandsänderungen über Bild- und Funktionstastatur
  - Netztrennstellen
  - Erdungspunkte

- Nachführung von Netzstrukturänderungen durch Ergänzung/Änderung der Datenbank
  - neue Umspannwerke, Schaltanlagen, Übergabestationen, Netzstationen
  - neue Meßwerte

- Netzübersichtsbild als Landkarte mit 4 Sammelmeldungen je Station

- 16 beliebig wählbare Meßwerte auf Meßwertschreiber aufzeichnen

- Abruf von Protokollen
  - Netztrennstellenübersicht
  - anstehende Warnmeldung usw.

- Gewährleistung sehr hoher Verfügbarkeit der Netzleitstelle

Alle diese Aufgaben wurden mit der in Bild 3 dargestellten Doppelrechnerkonfiguration gelöst. Die Tabelle 2 gibt eine prägnante Übersicht über die charakteristischen Daten des E-Netzes und der Automatisierungskomponenten.

Die gesamte Software der zentralen Netzleitwarte wurde mit dem BBC-PEARL-Programmiersystem realisiert. Sie umfaßt:

- ca. 60.000 PEARL-Statements
- ca. 0,5 M Byte für Programme
- ca. 5 M Byte Daten

Das Softwarepaket besteht aus 90 PEARL-Modulen und enthält:

- 39 Tasks
- 200 Prozeduren (auf Modulebene)
- 650 globale Namen

Zur Bearbeitung der Aufgaben sind folgende Datenspeicher notwendig:

- 150 K Worte Kernspeicher
  (davon ca. 20 K Worte für Betriebssystem)
- 7,5 M Byte Nachschubspeicher für Daten und Programme

Die Datenbasis des SMA-Systems zeigt das Bild 4. In den Bildern 5 und 6 sind am Beispiel der Aufgabe des überlagerten Netzschutzes die Bearbeitungsschritte, der Datenfluß und die globale Ablaufsteuerung dargestellt.

3. Erfahrungen mit dem Programmiersystem

In den gesamten, in Phasen aufgeteilten strukturierten Abwicklungsprozeß ist die Benutzung von PEARL und des PEARL-Programmiersystems eingebettet (s.Tabelle 3).

Der Rationalisierungseffekt bei Einsatz der Sprache beginnt bereits bei der Erarbeitung der Spezifikationen durch Benutzung der Datentypen und tritt voll in Erscheinung beim Entwurfsprozeß des Systems. Hierbei sind die PEARL-Konzepte für Tasking, Synchronisation und Dations sowie das der Blockstruktur (lokale und globale Daten) und des Parametermechanismus besonders tragfähig, um die zentrale Ablaufsteuerung darzustellen und Rückwirkungsfreiheit und Modularität der Programmbausteine zu sichern. PEARL-Elemente werden im Fortgang des Entwurfsprozesses als Pseudo-Code in Struktogrammen verwendet, so daß ein lückenloser Übergang bei der Codierung der PEARL-Module gewährleistet ist.

Die Qualität des BBC-PEARL-Programmiersystems prägt die sich anschließende Programmvorbereitungsphase. Beim Compilieren der PEARL-Module und dem Prelinken werden alle Formalfehler erkannt und prägnant ausgewiesen, so daß für Test und Inbetriebnahme das Aufspüren der inhaltlichen Fehler verbleibt. Dafür stehen quellenprogrammorientierte Test- und Bedienhilfsmittel zur Verfügung, d.h. ein Rückgriff auf die Assemblersprache ist nicht erforderlich.

Die vom Compiler und Prelinker erzeugten Statistiken erlauben, Optimierungen hinsichtlich Speicherraum und/oder Ausführungszeit zielsicher durchzuführen.

Änderungen und Erweiterungen des Programmpaketes waren durch die strukturierte Entwurfstechnik unter Ausschöpfung der Compiler- und Prelinkerlistings unproblematisch durchführbar.

Die Erfahrungen aus den abgewickelten Projekten zeigen, daß erst die Synthese aus einer strukturierten Projektabwicklung und einem der Sprache PEARL gerechtwerdenden PEARL-Programmiersystem, (d.h. eine echte PEARL-Umgebung), den optimalen Rationalisierungseffekt für Softwareerstellung und -pflege erreichen lassen. Diese Erkenntnis ist seither die Leitlinie der Projektabwicklung für die BBC-Netzführungssysteme.

1966: Beginn der Entwicklung von Sprache, Compiler und Betriebssystem für das Prozeß-Automatisierungs-System "PAS 1", basierend auf einem PL/I-Subset. Zielrechner war die "H 316".

Ab 1969: Einsatz dieses Software-Systems. PAS 1 war damals die erste, echte höhere Prozeßsprache, die in der BRD entwickelt wurde.

1970: Mitbegründung des PEARL-Arbeitskreises

1973: Definition eines Subsets aus dem Gesamtumfang der höheren Prozeßsprache PEARL 73. Entwicklung des zugehörigen Compilers, eines PEARL-Betriebssystems und eines Bedien- und Testsystems auf Sprachen-Niveau für die Prozeßrechnerfamilie "PDP 11":

- kleiner Subset
- kleines reaktionsschnelles Betriebssystem
- effizienter Compiler mit Compilezeit-Beeinflussung
- Testhilfsmittel, Bedienung

1974/75: erste Einsatzerfahrungen

- PEARL unterstützt die strukturierte Programmierung
- Vergrößerung des Subsets erforderlich
- Erweiterung des Betriebssystems ohne Verschlechterung der Reaktionszeiten
- Entwicklung von Hilfsmitteln für größere Projekte

1975 - heute: Erweiterung des BBC-PEARL-Programmiersystems

- PDP 11/03 .... PDP 11/70
- abgerundetes Subset
- ausgereiftes Gesamtsystem
- erprobt in mehr als 100 Projekten
- mit Platte oder reines Kernspeichersystem
- Wartung u. Pflege im industriellen Einsatz

Tabelle 1: Zeitlicher Ablauf des Einsatzes höherer Prozeßsprachen bei BBC

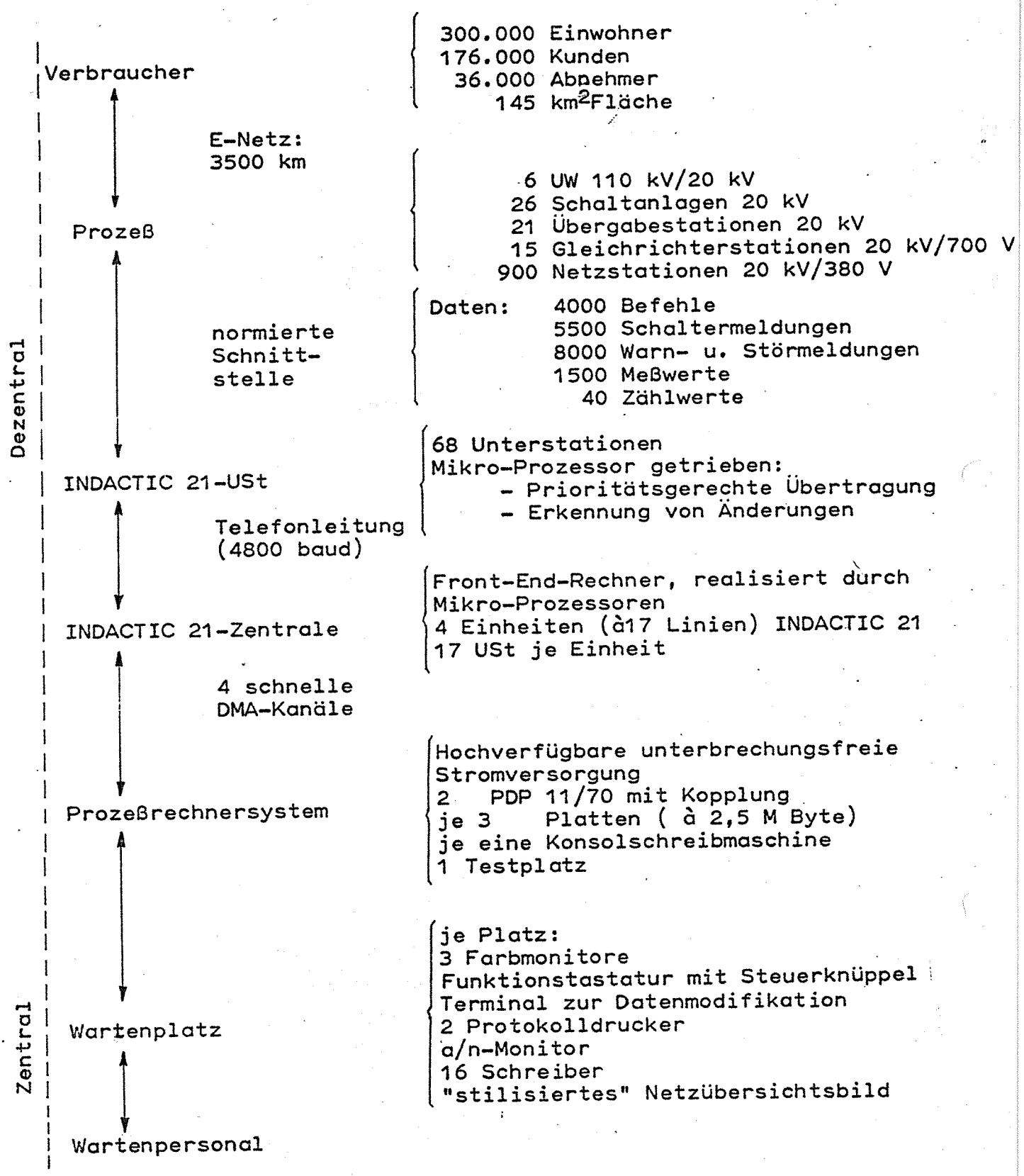


Tabelle 2: Charakteristische Daten des SMA-Systems

<table>
<tr><th>Projektphasen</th><th>PEARL-System</th><th>Einfluß</th></tr>
<tr><td rowspan="3">Aufgaben-definition

Spezifikation

Systementwurf grob

Systementwurf fein

Codierung

Übersetzung

Test

Inbetriebnahme

Pflege

Erweiterungen/ Änderungen</td><td></td><td rowspan="3">- Datentypen
- PEARL-Konzepte: Tasking, Synchronisation, Dations
- lokale, globale Daten
- PEARL als Pseudocode in Struktogrammen
- lückenloser Übergang auf PEARL-Module
- zügige Fehlerbeseitigung je Module
- Prelinker sichert Freiheit von Formalfehlern
- Problemlose Fehlersuche durch Test u. Bedienhilfs-mittel auf Quellenebene
- Optimierung zielsicher:
- Statisktiken d. Prelinken
- Compilezeit-Statement
- unproblematisch durch
- strukturierten Entwurf
- aktuelle Cpl-u. Prelinker-Listen</td></tr>
<tr><td>PEARL-Sprache</td></tr>
<tr><td>PEARL-Programmiersystem</td></tr>
</table>

Tabelle 3: Einfluß von PEARL bei den Projektphasen

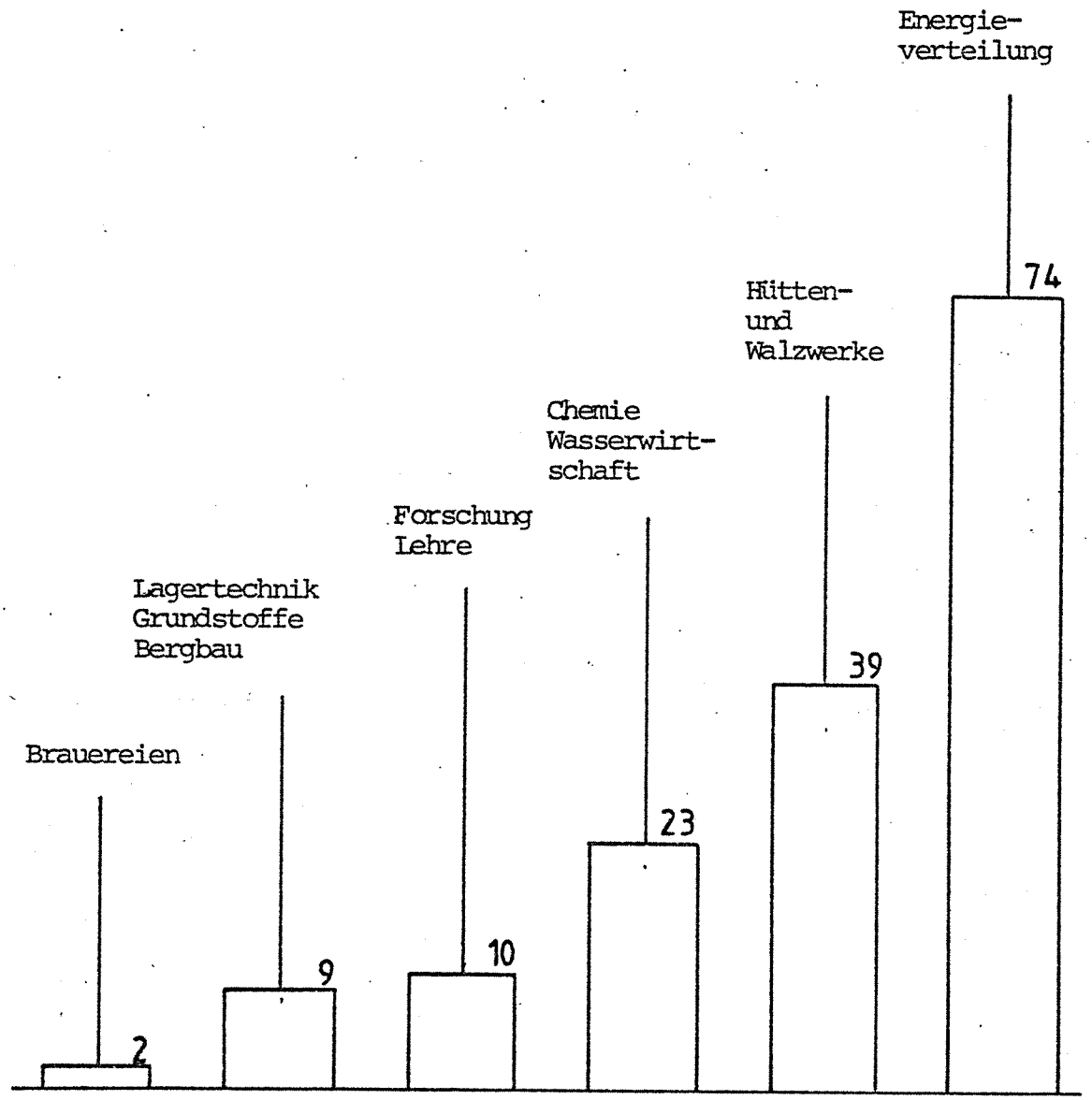


Bild 1: In Auftrag und Betrieb genommene PEARL-Rechner kumuliert.
Stand: Dezember 1979

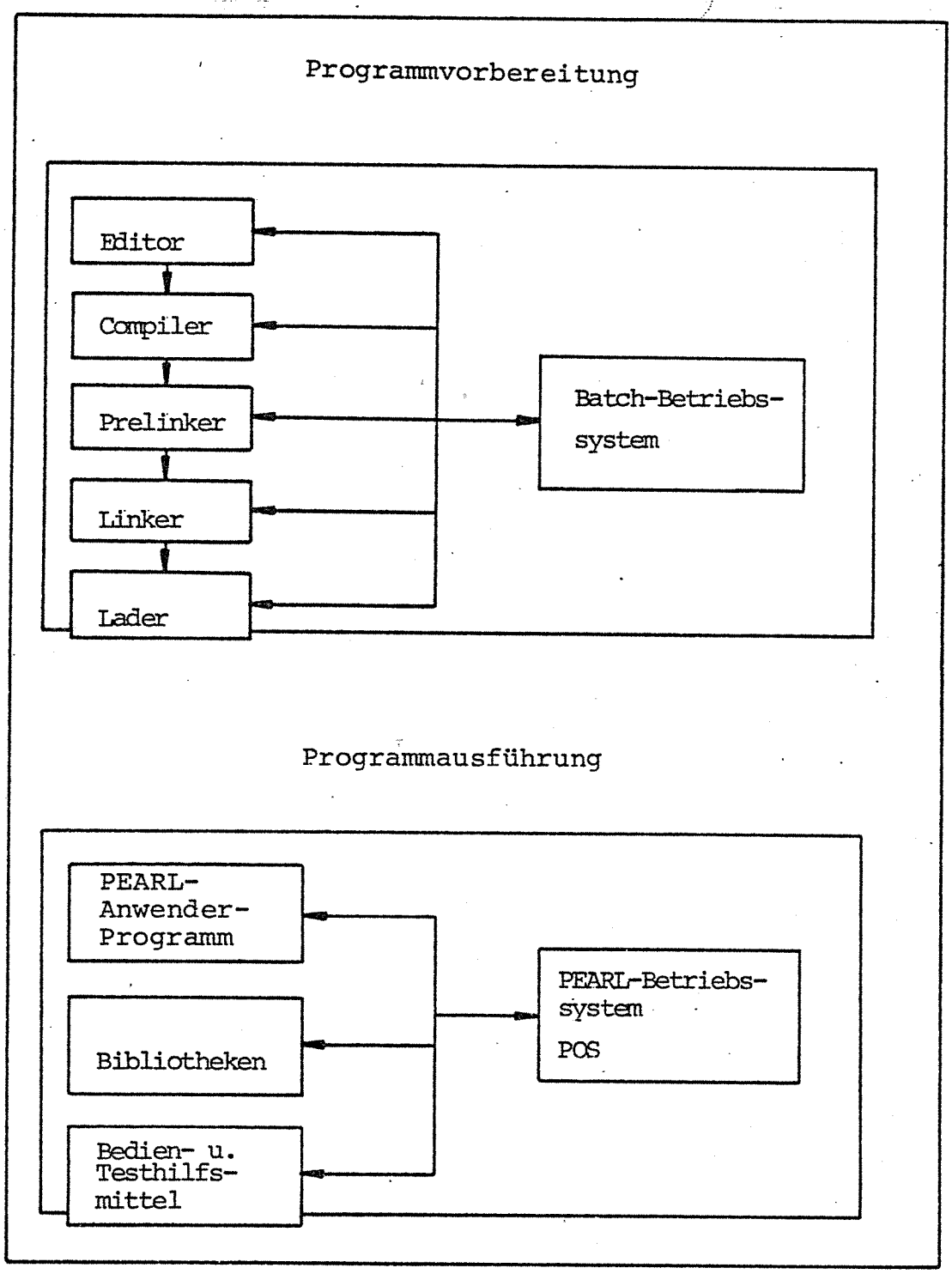


Bild 2: Komponenten des BBC-PEARL-Programmiersystems

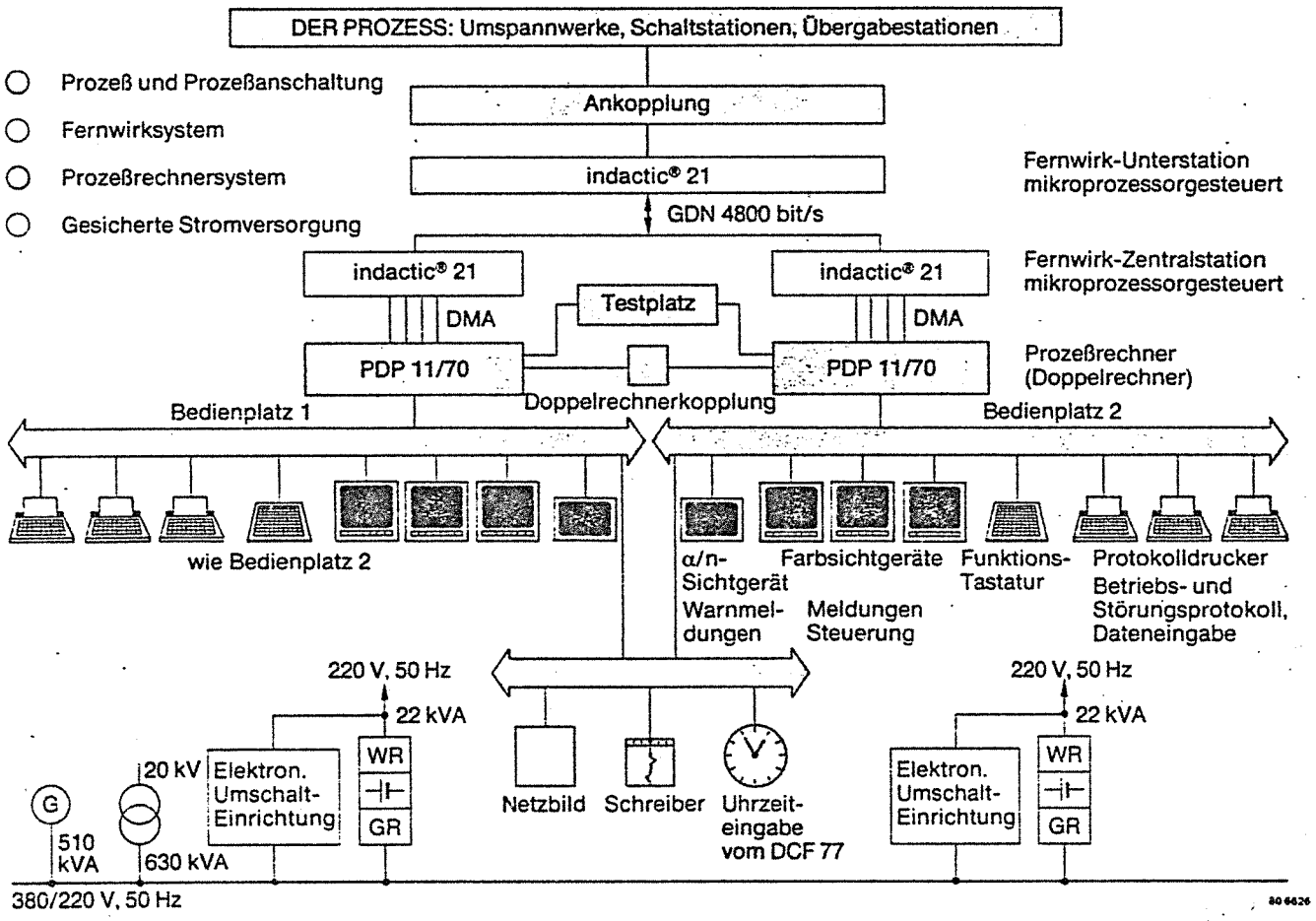


Bild 3: Blockschaltbild des Netzleitsystems für SMA

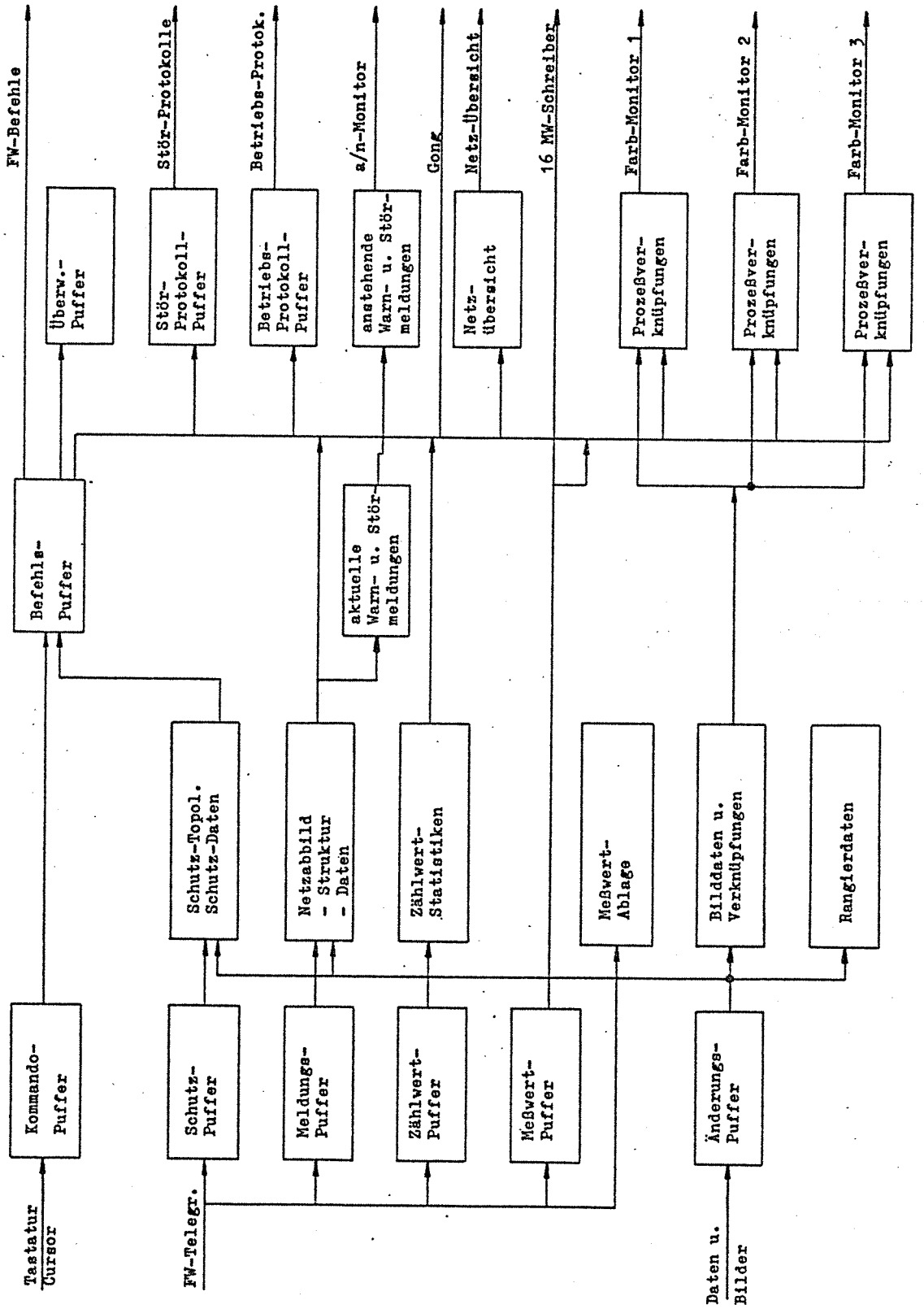


Bild 4: Datenbasis und Datenfluß des SMA-Systems

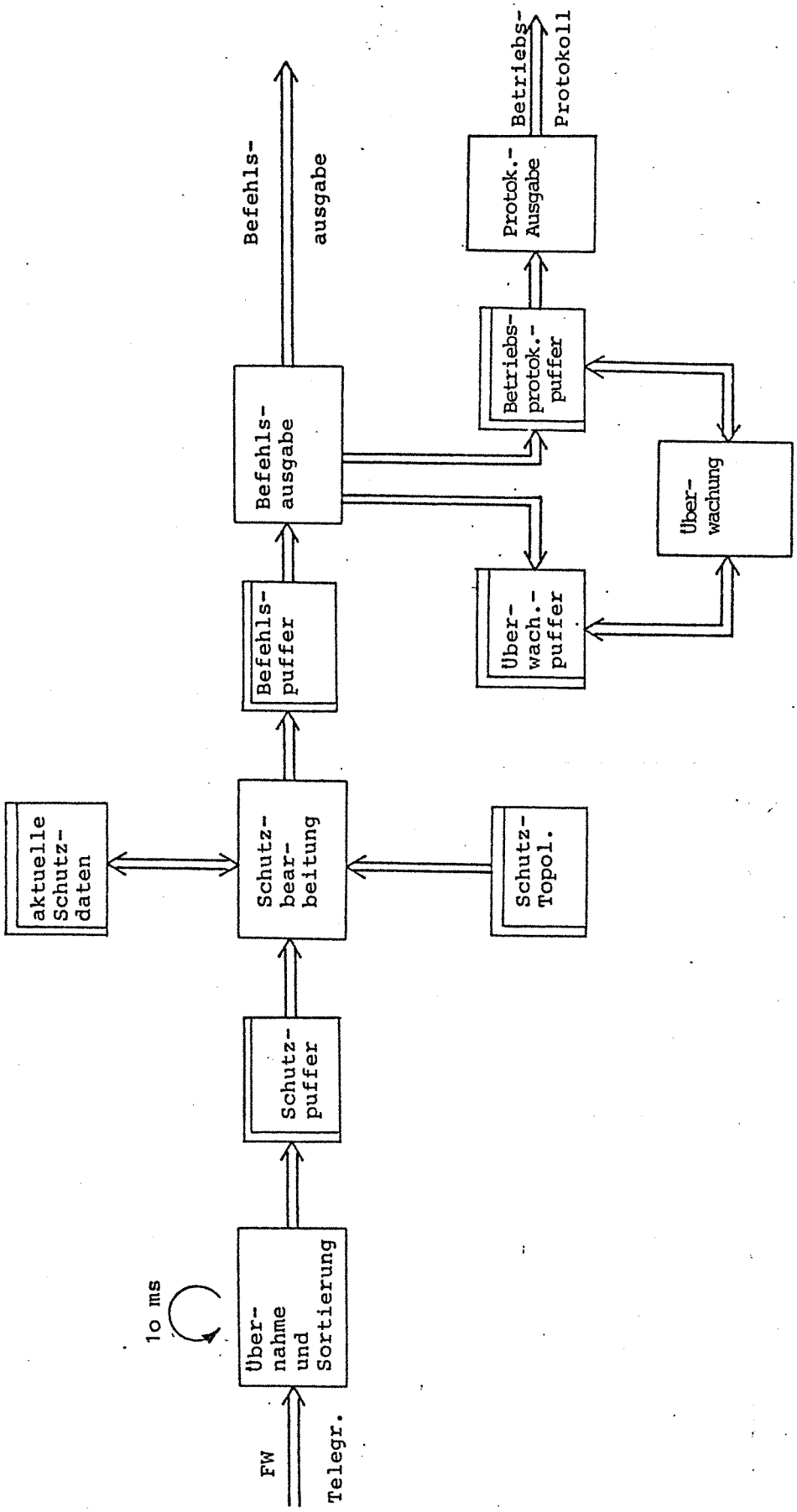


Bild 5: Bearbeitungsschritte und Datenfluß
für die Netzschutzaufgabe bei SMA

```
STARTM:   TASK; /*STARTTASK DES SYSTEMS */
          .
          .
          EVERY 10 MSEC ACTIVATE (TELEIN);
           /* EINPLANUNG DER ZYKLISCHEN ÜBERNAHME DER FW-TELEGRAMME*/
          .
          .
          END;
```

```
TELEIN:   TASK;  /* UEBERNAHME UND SORTIERUNG DER FERNWIRK-
                    TELEGRAMME */
          .
          .
          ACTIVATE  (SCHUTZ);    /* FALLS ANREGUNGS- ODER RICHTUNGS-
                                    KRITERIEN EINLAUFEN */
          .
          .
          END;
```

```
SCHUTZ:   TASK;  /* NETZSCHUTZBEARBEITUNG */
          .
          .
          ACTIVIATE (BEFAUS);  /* FALLS NETZSCHUTZFALL ERKANNT
                                  WIRD */
          .
          .
          END;
```

```
BEFAUS:   TASK;  /* AUSGABE VON SCHALTBEFEHLEN ZUR FERNWIRK-
                    PERIPHERIE */
          .
          .
          END;
```

Bild 6: Globale Steuerung für die Netzschutzaufgabe bei SMA

DAS PEARL-KOMPILIERSYSTEM FÜR DIE SIEMENS 300-16 Bit

Dipl.-Ing. K.F. Bamberger
Fa. Siemens AG - E STE 3 - 7500 Karlsruhe

Als sich im Jahre 1969 Vertreter von Herstellerfirmen, Instituten und Softwarehäusern trafen, um die Arbeit an einer Prozeßsprache aufzunehmen, war in unserem Hause die Forderung der Programmierer nach einer solchen Sprache sehr stark.

Die Prozeßrechner-Programmierer benutzten - mit wenigen Ausnahmen - Assemblersprache (Ausnahmen waren z.B. Prozeßmodelle oder Programme für Lastflußrechnungen, die in ALGOL 60 geschrieben wurden) und waren unzufrieden über das Ausprogrammieren immerwiederkehrender Routinen, die Fehlerquote bei der Notierung, die schwere Verständlichkeit der Protokolle und die Mühe, die es machte, sich in Programme anderer hineinzufinden, diese zu ändern bzw. zu ergänzen.

Wir hatten deshalb bereits begonnen, Unterprogramm- oder Makrobibliotheken aufzubauen - Makroassembler waren die logische Konsequenz - wir arbeiteten eine Assemblernotation aus, die insbesondere für Adreßrechnung und arithm. Verknüpfungen an Höhere Sprachen (FORTRAN) heranreichte und heranreicht, FORTRAN wurde um Prozeß-Calls erweitert, zusätzlich entstanden die technol. orientierten Standardsysteme (MADAM, SOSYNAUT, CEPAMAT, SIMAT und eine Reihe anderer).

Dieser sehr kurze Rückblick sollte einerseits zeigen, daß die Suche nach leistungsfähigen Programmerstellungsmitteln und damit auch nach einer Prozeßsprache bereits damals im vollen Gange war, andererseits aber auch die Umgebung umreißen, in die die Sprache PEARL, zumindest bei uns, hineingestellt werden mußte und in der sie sich auch heute noch behaupten muß.

Die diesem Bericht zugrunde liegenden Arbeiten wurden mit Mitteln des Bundesministers für Forschung und Technologie gefördert. Die Verantwortung für den Inhalt liegt jedoch allein beim Autor.

Siemens hat also von Anfang an aktiv an der PEARL-Entwicklung mitgearbeitet und 1976 einen Pilotcompiler fertiggestellt, der sich an den bis dahin veröffentlichten PEARL-Sprachmitteln orientierte. Diese Pilotentwicklung machte es uns dann auch möglich, bereits im Januar 1978 einen Basis-PEARL-Compiler für den Prozeßrechner PR 330 freizugeben, obwohl erst im September 77 der einzig verfügbare, allgemein anerkannte Subset von Basis-PEARL verabschiedet worden war bzw. erschienen war. (Bild 1).

Das Siemens-PEARL-Kompiliersystem

Zunächst möchte ich einige Entwurfs- und Realisierungskriterien des PEARL-Kompiliersystems, die die Struktur des Compilers wesentlich beeinflußt haben, auflisten.

1. Integration des PEARL-Compilers in die vorhandene Systemprogramm-Umgebung:
   - Ablauf des PC unter den Standard-Organisationsprogrammen der Prozeßrechnersysteme 300-16 Bit
   - Einhalten der für Systemprogramme geltenden Bediensyntax
   - Verwenden vorhandener Dienst- und Hilfsprogramme wie Binder, Lader, Monitor
   - Archivierung, Compilerproduktion und Fehlerdiagnose mit bewährten Verfahren

Damit ist auch zum Ausdruck gebracht, daß es nicht genügt, nur einen Compiler zu realisieren, sondern daß das Vorhandensein einer entsprechend leistungsfähigen Systemprogrammumgebung mindestens genau so wichtig ist.

2. Ein PEARL-Programm muß sich wie jedes andere Anwenderprogramm verhalten:
   - Ablauf unter Standard-Organisationsprogrammen
   - Zusammenwirken mit Nicht-PEARL Programmen oder Programmpaketen

3. Keine Unterscheidung zwischen Übersetzungs- und Zielrechner, d.h. eine Kompilierung muß sowohl im Rechenzentrum als auch vorort on-line möglich sein:
   - Einhaltung eines vorgegebenen Laufbereiches
   - Geringe Transferbelastung zum Peripherspeicher (solche Transfers bilden i.a. einen Engpaß bei Prozeßaufgaben)

4. Wirkungsvolle Unterstützung beim Testen von PEARL-Programmen:
   - Leistungsfähige Testhilfen zur Laufzeit
   - Prägnante Fehlertexte im Kompilierprotokoll am Ort des Fehlers
   - Hohe Kompiliergeschwindigkeit

   Die Bilder 2 u. 3 sollen diese Aussagen noch etwas mehr veranschaulichen.

Die Forderung nach einem portablen Compiler hatten wir nicht und zwar, weil wir aus den Untersuchungen am Pilotcompiler sicher zu sein glaubten, Forderungen wie:

> Schnelle Übersetzungszeiten und Compiler als Systemprogramm (z.B. Laufbereich)

mit einem solchen portablen Compiler nicht zusätzlich erfüllen zu können.

Bild 4 soll schematisch aufzeigen, mit welchen Mitteln erreicht wurde, diese z.T. widersprüchlichen Forderungen einer befriedigenden Lösung zuzuführen.

Die naheliegende Lösung ist ein Mehrpaßcomiler, wobei umfangreiche logisch einheitliche Aufgaben und mehrere Pässe (z.B. Syntax-Analyse auf Pässe 2, 3 und 5) verteilt wurden.

Ein solcher Mehrpaßcompiler erfordert allerdings eine Reihe zusätzlicher Transfers, was der Forderung nach hoher Compiliergeschwindigkeit zuwiderläuft.

Deshalb wurden so wenig wie möglich Pässe gewählt und darüberhinaus wurden logisch getrennte Aufgaben in ein und demselben Paß untergebracht (hier z.B. im Paß 5 die Anweisungsanalyse und die Erzeugung der umgek. Poln. Nation).

Ein weiterer Schritt in Richtung hohe Kompiliergeschwindigkeit war darauf gerichtet, eine möglichst geringe Belastung der Peripherspeichernahtstelle zu erreichen, d.h. die erforderlichen Plattentransfers auf ein notwendiges Minimum zu begrenzen. Dies wurde u.a. mit folgenden Maßnahmen erreicht:

- Absetzen von kurzen Zwischen-Codes in den einzelnen Pässen, durch Einträge variabler Länge (also eine kompakte Zwischensprache)
- Möglichst viel Informationen in sequentiell zu lesendem Zwischencode bearbeiten - also bearbeiten und vergessen! und nicht über Tabellen
- Einrichten eines eigenen Passes für den Listenaufbau, um einen großen Hauptspeicher-Listenbereich zu erhalten
- Den HSP-Listenbereich flexibel gestalten, um eine möglichst gute Ausnutzung des vorhandenen Laufbereiches zu erreichen
- Die Syntaxanalyse weitgehend sackgassenfrei ausführen, womit erreicht wird, daß der Input-Zeiger selten auf einen schon gelesenen Eintrag zurückzusetzen ist.

Schließlich wurden, um der Forderung "hohe Kompiliergeschwindigkeit und Einhalten eines Laufbereichs von max. 20 KW" möglichst nahe zu kommen, die Sprachen MECO und Assembler zur Implementierung benutzt.

MECO ist eine für Compilerbau, genauer für Syntaxanalyse, im Hause Siemens bereits mehrfach eingesetzte und bewährte Sprache. Als Ergebnis eines Kompiliervorganges geht es dem Anwender u.a. auch darum, exakte Hinweise auf den Ort eines Fehlers, verbunden mit einem aussagekräftigen Fehlertext für syntaktische Fehler, zu erhalten.

Bild 5 zeigt die Realisierung: Es werden, wo immer es geht, Fehlermeldungen im Quellspracheprotokoll am Ort des Fehlers eingefügt. Die erwähnten Fehlermeldungen erscheinen automatisch im Kompilierprotokoll solange Syntaxfehler vorhanden sind.

Für den Laufzeittest oder logischen Test kann man zusätzlich im Testmode übersetzen, d.h. man führt in diesem Fall eine zusätzliche Bedienung aus und erhält als Ergebnis ein PEARL-Adreßbuch, und im Grundspracheprotokoll wird zusätzlich Testhilfeinformation eingefügt (Bild 6). Letztere beinhaltet z. B. Zeilennummern, Informationen (Aufrufe) für das Testsystem, während das Adreßbuch die Variablennamen mit Adresse (blockspezifisch) enthält.

Mit der so erhaltenen Zusatzinformation und einem PEARL-Testsystem, das beim Binden der Grundsprache hinzugefügt wird, können im Dialog z.B. folgende Testfunktionen ausgeführt werden:

- Protokollierung durchlaufener Quellsprachezeilen, also ein Trace, und das Anhalten am Beginn vorwählbarer Zeilen
- Anhalten, wenn eine Variable einen voreingestellten Wert erreicht hat und
- Manipulation von Variablen mit Hilfe des im Prozeßrechnersystem 300-16 Bit enthaltenen Testsystems TEPOS (TEPOS im Bild nicht dargestellt).

Diesen Compiler, den ich Ihnen mit seinen charakteristischen Merkmalen und in seiner Systemprogrammumgebung kurz vorgestellt habe, wurde, wie bereits erwähnt, im Januar 78 freigegeben, er ist also seit nunmehr 2 1/2 Jahren im Einsatz.

In dieser Zeit haben wir 3 weitere Freigaben dieses Compilers herausgebracht (Bild 7), so daß nunmehr die Variante D vorliegt.

Es kann mehrere Gründe für weitere Freigaben eines SW-Produktes geben. Fehlerbeseitigungen, Funktionserweiterungen und/oder Verbesserungen sind wohl die häufigsten.

Da jedes Systemsoftwareprodukt einer gründlichen Abnahmekontrolle vor Freigabe unterworfen wird, wäre die Zahl der bisher aufgetretenen Fehler kein Grund für 3 weitere Freigaben gewesen, man

hätte das durch Nachführen beheben können. Funktionserweiterungen in Form von Spracherweiterungen haben wir nicht vorgenommen, da für die Akzeptanzererprobung der Basis-Subset uns ausreichend erschien. So haben wir den Schwerpunkt auf Verbesserungen, d.h. Optimierung des Compilers, gelegt.

Variante B - Umstellung auf R-Familie

- Feldzugriff: Ausnutzung der HW für wortbündige Feldelemente (z.B. FIXED, FLOAT) durch spezielle Codeerzeugung und Optimierung der Adreßberechnung für ein Feldelement.

Variante C - Laden des Laufzeitsystems in Common DATA-Bereich (ablaufinvariantes Laufzeitsystem)

- Binäre E/A: Ausnutzung des Wortrasters bei der Aufbereitung des E/A-Puffers und die Zusammenfassung von Verbunden (Extremfall ist Bit-Feld zu mehreren Worten à 16 Bit

Variante D - Adreßraumerweiterung: z.B. virtuelle Adressierung (PEARL C, d.h. globale Anwenderdaten in eigenem Laufbereich)

- Überarbeitung des gesamten Laufzeitsystems, z.B. unter Verwendung der mit der 300 R-Familie zur Verfügung stehenden HW-Befehle wie doppelt lange-, Gleitpunkt-, Byte-Befehle.

Bild 8 zeigt an Hand von zwei unterschiedlichen Beispielen die mit diesen Optimierungen erzielten, gemessenen Laufzeitverbesserungen.

Beispiel 1 Es handelt sich um ein rechenintensives Programm aus der Lastflußrechnung (eine gedrängt gespeicherte - da schwach besetzte - Matrix aus komplexen Zahlen wird invertiert).

Variante A lieferte gegenüber der ausprogrammierten Assemblerlösung ein um ca. 2,3-fach langsameres Ergebnis. Die Optimierung des Feldzugriffes reduzierte

diese Laufzeit auf den Faktor 1,4, und die des Laufzeitsystems (hier besonders die Routine zur Adreßberechnung eines Feldelements) reduzierte nochmals und ergab nun nur noch eine Verlangsamung um den Faktor 1,18 gegenüber der Assemblerlösung.

Beispiel 2 Binäres Transferieren eines 256 W langen Feldes:
Dieses Beispiel zeigt, daß nach der Optimierung der binären E/A die Transferzeit praktisch nur durch die mittlere HW-Zeit bestimmt wird;mittlere HW-Zeit:
= mittlere Zugriffszeit, Org-Zeit, eigentlicher Transfer.

Diese Ergebnisse sind jedoch nicht zu verallgemeinern. Ich möchte ausdrücklich betonen, daß diese äußerst positiven Ergebnisse nicht für alle Anwendungen zutreffen. So wirkt z.B. die Optimierung im letzten Beispiel dann nicht, wenn man skalare Größen transferiert.

Dennoch glauben wir, durch Einarbeiten eigener und Anwendererfahrungen inzwischen einen äußerst stabilen Compiler zu haben, der den gesteckten, zu Anfang meines Referates aufgezeigten Zielen weitgehend entspricht. (Bild 9).

Die Energieversorgung Ostbayerns, OBAG, war nun ihrerseits bereit, ein Pilotprojekt unter Verwendung von Basis-PEARL zu starten und die Tragfähigkeit der Sprache, aber auch die Leistungsfähigkeit unseres PEARL-Kompiliersystems zu erproben. Das PEARL-Kompiliersystem wurde hier von Anfang an eingesetzt und einige wesentliche Verbesserungen - z. B. die von mir gezeigte Transferoptimierung - wurden aufgrund der hier gemachten Erfahrungen durchgeführt und es hat sich bestätigt, daß sowohl die Sprache als auch die Compiler im Einsatz reifen müssen.

Ich möchte den späteren Ausführungen zwar nicht vorgreifen, doch darf ich schon soviel sagen: die gemachten Erfahrungen mit Basis-PEARL sind insgesamt sehr positiv und berechtigen dazu, den eingeschlagenen Weg fortzusetzen.

1969	Beginn der Sprachentwicklung
7/76	Fertigstellung eines Pilot-Compilers bei SIEMENS
9/77	Beschreibung von Basis-PEARL
1/78	Freigabe des ersten Basis-PEARL-Compilers durch SIEMENS, für Rechner SIEMENS 330

Bild 1 Entwicklung des PEARL-Compilers

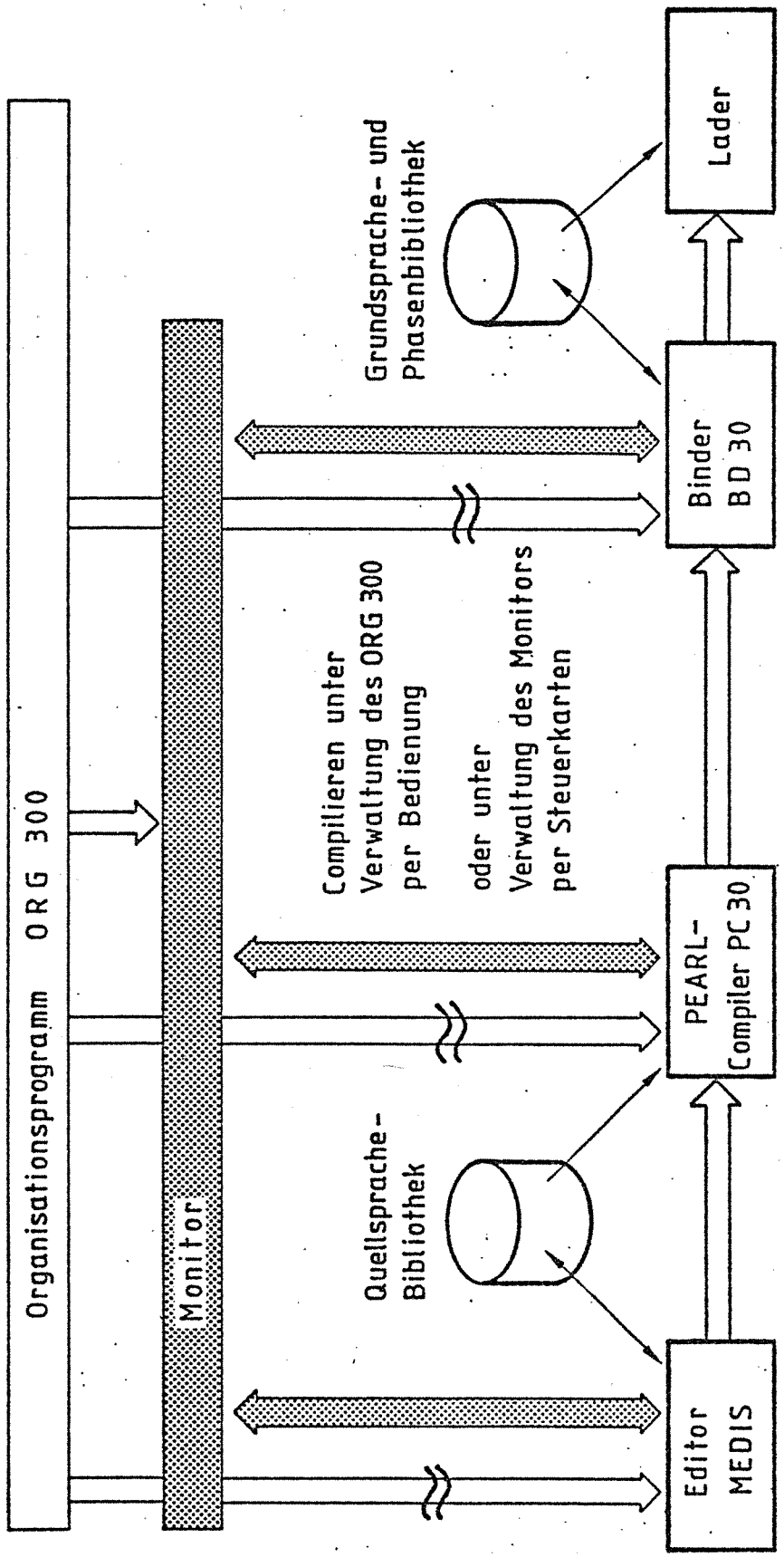
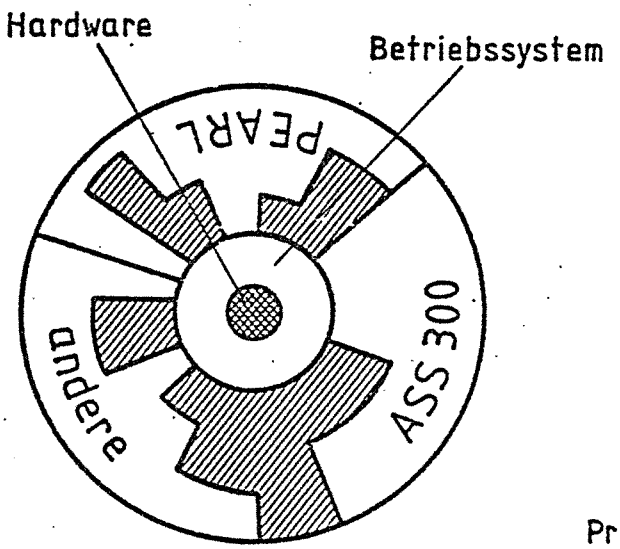
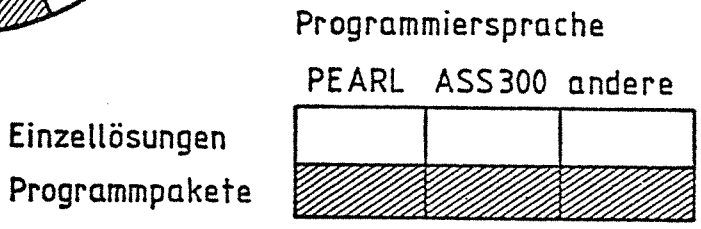


Bild 2 PEARL-Compiler im Spektrum der Dienstprogramme





PEARL-Programme
und
<u>andere Anwenderprogramme</u>
müssen

- sich gegenseitig starten
- sich gegenseitig koordinieren
- Daten austauschen
- gemeinsame Geräte benutzen
- sich bezüglich z.B. Wiederanlauf udgl. ähnlich verhalten

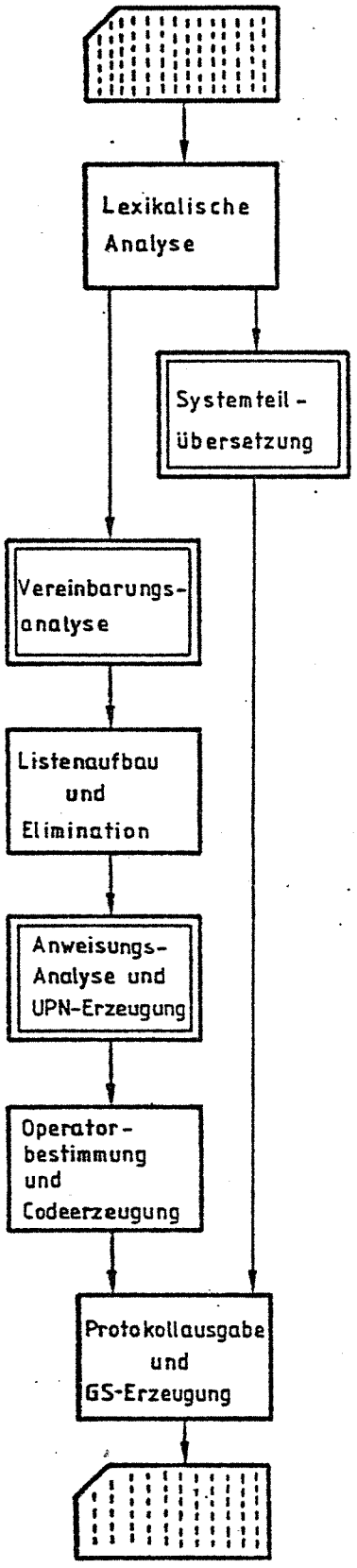
daher ⇒ Kein eigenes Betriebssystem für PEARL

PEARL-Programme benutzen ORG 300

PEARL-Programme
und
<u>Programmpakete</u>
müssen darüber hinaus

- über technologisch orientierte Datennahtstellen verkehren

Bild 3 Zusammenwirken PEARL-Programm-Umgebung



- Mehrpaßcompiler

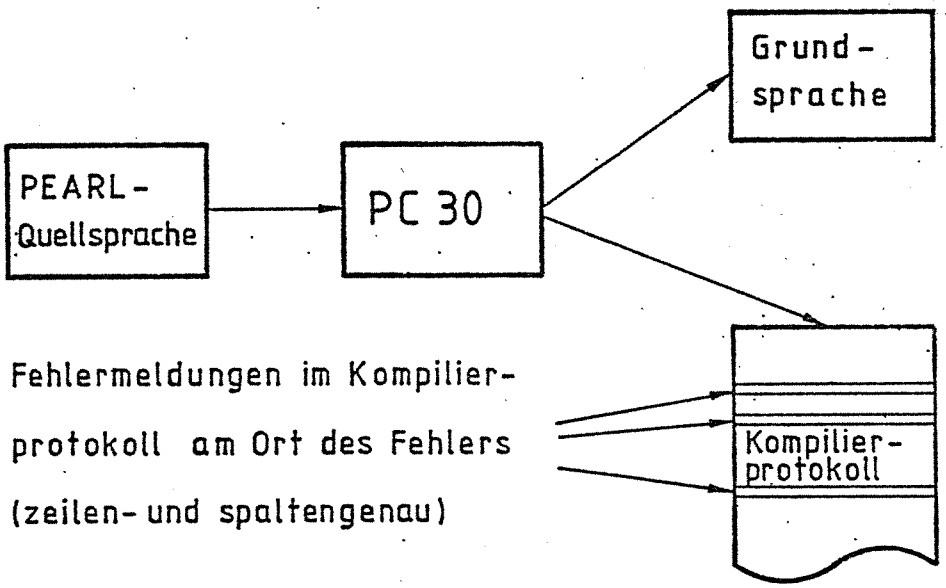
Trotzdem gute Kompiliergeschwindigkeit durch

- Wenig Pässe
- Geringe Belastung der Peripherspeichernahtstelle
- Geeignete Implementierungssprachen

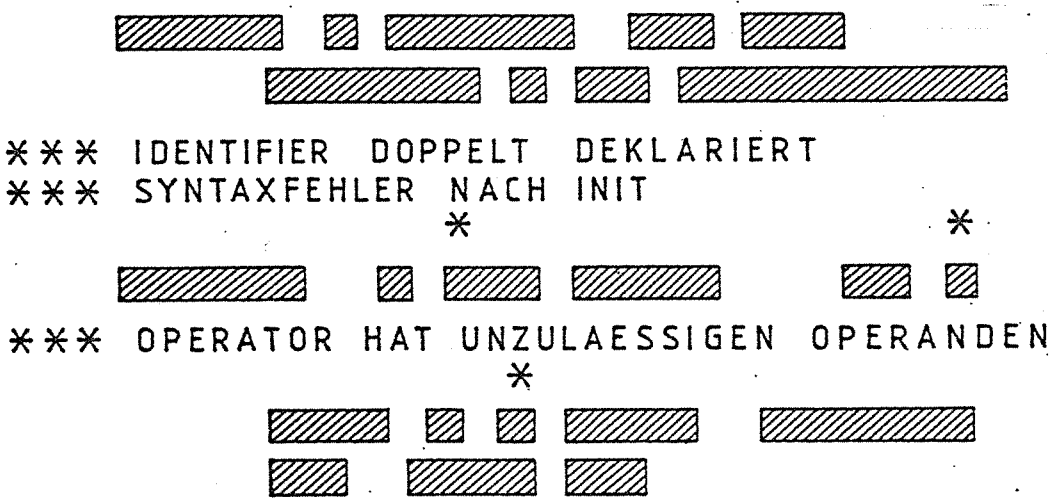
MECO für Syntaxanalyse

Assembler

Bild 4 Aufbau des PEARL-Compilers



Beispiel für Fehlermeldungen (schematisch)

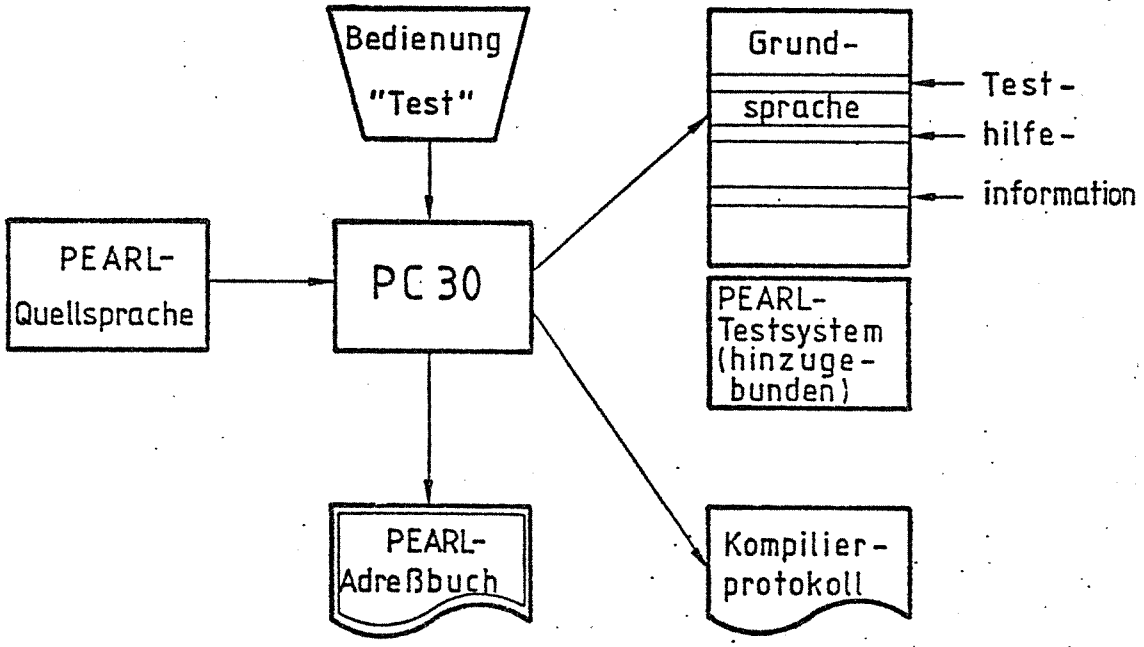


PEARL-Quelltext

* * * Fehlermeldezeilen (* in Folgezeile bezeichnet Spalte)

Bild 5 Fehlermeldungen des PEARL-Compilers

Übersetzen im Testmode



Testen im Dialog

Stufe 1: Mit Kompilierprotokoll und PEARL-Testsystem

- Trace (Protokollierung der Nummern durchlaufener Zeilen)
- Haltepunkte setzen (an Zeilenanfängen)

Stufe 2: Mit PEARL-Adreßbuch und PEARL-Testsystem

- Variablenkontrollstop (wertabhängig)

Stufe 3: Mit PEARL-Adreßbuch und Assembler-Testsystem TEPOS (nicht im Bild)

- Werte von Variablen überprüfen/ändern

Bild 6 Hilfsmittel für den Laufzeittest

Compiler seit $2\frac{1}{2}$ Jahren im Einsatz:

Variante	Besonderheit	Freigabe	Optimierung
A	für 330	1/78	
B	für R-Serie	1/79	Feldzugriff
C	Laden des Lauf-zeitsystems in CD	11/79	Binäre E/A
D	Adreßraumer-weiterung	4/80	Laufzeitsystem als Ganzes

Bild 7 Varianten des PEARL-Compilers

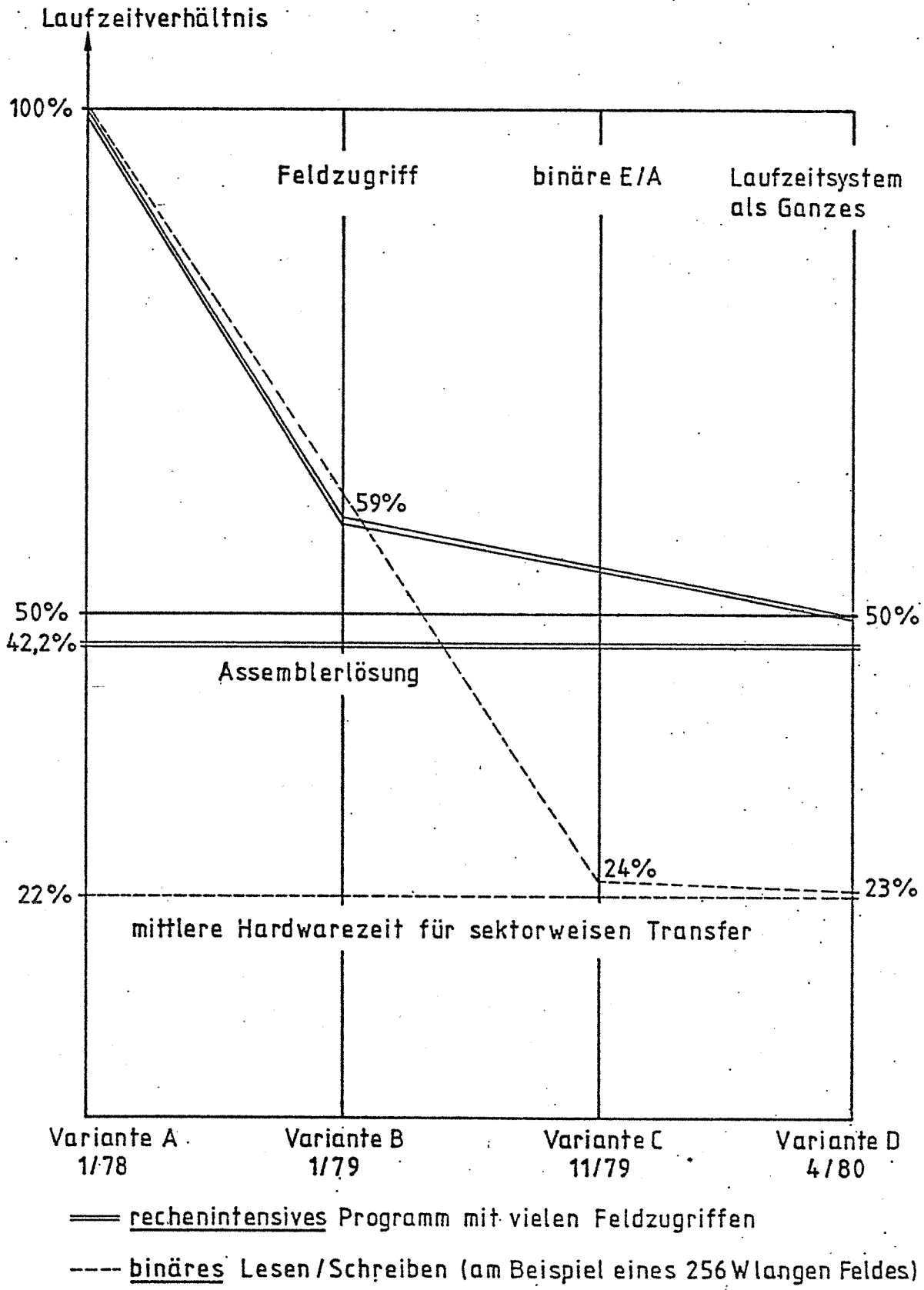


Bild 8 Laufzeitoptimierung

- Platzbedarf auf Platte

Ausgelieferter PC 30	190 kW
Ausgelieferte Laufzeitroutinen	65 kW
Vom Ladebinder erstellte Version (einfache Listenlänge: Page = 1)	195 kW

- Platzbedarf im HSP

Laufbereich für Compiler	≥ 17 kW
Generierbares ablaufinvariantes Laufzeitsystem	≤ 17 kW

- Lieferform

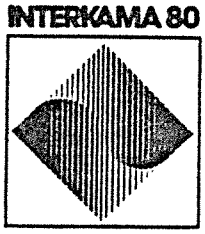
  Auf Platte, einzeln oder innerhalb Systempaket

- Übersetzungsgeschwindigkeit

  ~ 350 - 700 Zln/min

  (~ 4 kW Code/min)

Bild 9 Technische Daten des PEARL-Compilers



PEARL
ASSOCIATION
VEREIN E.V.

Der PEARL-Verein ist auf der INTERKAMA - Halle 6 - auf dem Stand des KfK beim Projekt PFT vertreten.

Wir bieten an:

Informationen über

- Spracheigenschaften von PEARL
- Verfügbarkeit von PEARL
- Anbieter von PEARL

und ein Preisausschreiben, bei dem Sie Bücher über die Prozessrechnersprache PEARL gewinnen können.

Besuchen Sie uns auf der INTERKAMA 80!

PEARL-Tagung 80

am 8.-9. Dezember 1980
in Düsseldorf
VDI-Haus

Motto:
PEARL in der
Anwendung

PROGRAMMAUSSCHUSS:

Dr. P. Elzer	DORNIER-System
Prof. Dr. F. Hofmann	Universität Erlangen
Dr. Hommel	BMFT
Dipl.-Math. G. Koch	BBC
Prof. Dr. H. Krüger	Universität Karlsruhe
Prof. Dr. R. Lauber	Universität Stuttgart
Dr.-Ing. U. Mayr	OBAG

WISSENSCHAFTLICHE TAGUNGSLEITUNG UND ANMELDUNGEN:

Dr.-Ing. U. Mayr
Energieversorgung Ostbayern AG
Prüfeninger Straße 20
8400 Regensburg
Tel. 0941/201 403

TAGUNGSBEITRAG:

Voller Beitrag	95,-- DM
GI- und GMR-Mitglieder	70,-- DM
Mitglieder PEARL-Verein	45,-- DM
Studenten	15,-- DM

WICHTIG:

Aus Platzgründen ist die Teilnehmerzahl auf 150 begrenzt.
Die Berücksichtigung erfolgt in der Reihenfolge der Anmeldung.

PROGRAMM

MONTAG, 8.12.1980

11^{00} : BEGRÜSSUNG
R. Lauber, U. Mayr

11^{15} : PEARL FÜR MIKROPROZESSORENVERBUND
Sitzungsleitung: U. Mayr

H.-J. Schneider, DORNIER-System
PEARL-Softwaresystem für gekoppelte Klein- und Mikrorechner

H. Steusloff, IITB
PEARL für Mehrrechnersysteme - Erfahrungen und notwendige Konsequenzen

H. Drtil, BGT / K. Lukas, GPP
PEARL-System für 8086 mit potablem Cross-Compiler

13^{15} : MITTAGSPAUSE

14^{30} : ANWENDUNGEN IN DER ENERGIEVERSORGUNG
Sitzungsleitung: G. Hirschberg

D. Struhalla, Siemens
Erfahrungen mit Basic-PEARL im Projekt Netzleitstelle Deggendorf

K. Maurer, BASF
Einsatz des BBC-PEARL-Subsets bei der Realisierung eines Prozeßinformationssystems zur Oberwachung des Fremdstrombezugs eines chemischen Werkes

16^{00} : KAFFEEPAUSE

16^{30} : ANWENDUNGEN IN DER WEHRTECHNIK
Sitzungsleitung: R. Bodem

K. Ziemlich, Batelle-Institut
Einsatz von PEARL bei der Softwareentwicklung für eine FlaSchießplatz-Automatisierung

E. Fahr, DORNIER-System
PEARL am Beispiel eines Ausbildungssystems

18^{00} : ENDE

20^{00} : MITGLIEDERVERSAMMLUNG DES PEARL-VEREINS

DIENSTAG, 9.12.1980

9^{00} : ANWENDERSYSTEME
Sitzungsleitung: R. Lauber

K. Goede, ADV/ORGA
Datenhaltung mit PEARL

E. Eiben, ADV/ORGA / F. Ruland, F.A. Meyer GmbH
Simulation von Prozeßdaten für PEARL-Programm-Systeme

10^{30} : KAFFEEPAUSE

11^{00} : ERFAHRUNGEN MIT PEARL
Sitzungsleitung: G. Koch

T. Krumnak, Krupp Atlas-Elektronik
Anwendungserfahrungen mit einer PEARL-Implementation auf einem Kleinrechner

A. Storr, Universität Stuttgart
Erfahrungen mit PEARL bei der Erstellung von Programmen zur Steuerung und Oberwachung von flexiblen Fertigungssystemen

12^{30} : MITTAGSPAUSE

14^{00} : PEARL IM VERGLEICH MIT ANDEREN PROGRAMMIERSPRACHEN
Sitzungsleitung: P. Elzer

H. Rzehak, Hochschule der Bundeswehr / G. Hommel, BMFT

15^{30} : ABSCHLUSSDISKUSSION

16^{00} : TAGUNGSENDE