PEARL Rundschau

Inhalt

Vorwort 1

H. Mittendorf
Erfahrungen mit PEARL in Firmenschulung und Hochschulvorlesungen 3

H. Windauer
Erfahrungen mit PEARL Schulungen im industriellen Bereich 11

E. Fahr
Anwendererfahrung mit PEARL bei Kursen und nachträglichem Einsatz in Projekten 14

T. Hänssgen
Ausbildungsplan für die PEARL-Schulung bei Digital Equipment und erste Erfahrungen 18

L. Frevert
PEARL in Vorlesungen und Praktika der FH Bielefeld 26

B. Reißenweber
PEARL Ausbildung im Studiengang Elektrotechnik an der Universität (GH) Paderborn 30

T. Roehrich
Erfahrungen, Konzept und Ablauf verschiedener PEARL-Schulungen am IVD Stuttgart 36

Überblick über Lehr- und Ausbildungsveranstaltungen 42

Presseschau 44

Leserbriefe 46

November 1981 Band 2 Nr. 5

PEARL Rundschau

November 1981
Band 2
Nr. 5

Der PEARL-Verein e.V. (PEARL-Association) hat das Ziel, die Verbreitung der Realzeitprogrammiersprache PEARL (Process and Experiment Automation Realtime 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.

Herstellung durch die VDI-Verlag GmbH, Düsseldorf

Die PEARL-Rundschau ist das Mitteilungsblatt des PEARL-Vereins e.V. Neben Vereinsangelegenheiten werden für alle PEARL-Interessenten Informationen über Erfahrungen, Anwendungsmöglichkeiten und Produkte gegeben.
Preis des Einzelheftes für Nichtmitglieder: DM 15,—, für Studenten DM 5,—. Jahresabonnement DM 75,—.
Zur Veröffentlichung werden sowohl fachliche Abhandlungen über Realzeit-Anwendungen, als auch Kurznachrichten zu Themen des Prozessrechnereinsatzes und der Verwendung höherer Sprachen angenommen, soweit sie für PEARL-Interessenten von Bedeutung sein können.

Es obliegt dem Autor, die Rechte zur Veröffentlichung seines Beitrags in der PEARL-Rundschau sicherzustellen. Der PEARL-Verein erhebt keine Einwände gegen das Kopieren einzelner Beiträge bei Angabe der Quelle.

Beiträge können jederzeit, auch unaufgefordert, an ein Mitglied des Redaktionskollegiums geschickt werden. Autoren werden gebeten, bei der Schriftleitung ein Info-Blatt zur Manuskriptgestaltung anzufordern.

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 zur 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

Dies ist also das zweite Heft zur Tagung "PEARL in Ausbildung, Lehre und Schulung", die am 1./2.10. in Erlangen stattfand. War der erste Teil rein auf den Bereich Hochschulen und Fachhochschulen beschränkt, so steht diesmal mit vier Beiträgen der industrielle Bereich im Vordergrund. Dabei wird doch deutlich, daß hier manchmal etwas andere Gesichtspunkte wesentlich sind, als im universitären Bereich. So muß sich PEARL eben auch der Konkurrenz in Form anderer Programmiersprachen stellen. Etwas Anderes fällt noch bei der Lektüre der Artikel in diesem Heft auf: auch bei den Schulungen außerhalb der Industrie waren hier weniger die "didaktisch-analytischen" Gesichtspunkte wesentlich, sondern die Möglichkeit, durch praktische Übungen ein direkteres Verhältnis zum Gelernten zu bekommen. Die gute alte "hands-on experience" ist eben durch nichts zu ersetzen.

Von besonderem Interesse für potentielle PEARL-Anwender sollte auch die Übersicht sein, in der dargestellt wird, wo PEARL überall gelehrt wird. Man kann darauf z. B. entnehmen, wo man eventuell geschulte Mitarbeiter findet.

Eine weitere erfreuliche Neuigkeit dürfte diesmal die Presseschau sein. Das Presseseminar und die inzwischen angelaufene kontinuierliche Pressearbeit scheinen Früchte zu tragen. Vielleicht setzt sich doch noch die Einsicht durch, daß das Heil nicht nur von über dem Ozean kommt. Es ist dem Unterzeichneten sowieso immer unverständlich gewesen, daß ein Land, das so von fortgeschrittener Industrie und Export abhängt, wie das unsere, sich derartig ins Bockshorn jagen läßt, was die wichtige Schlüsseltechnologie "Datenverarbeitung" angeht.

Als letztes ein weiteres Novum: der erste Leserbrief ! Eigentlich müßte so etwas gebührend gefeiert werden. Hoffentlich folgen bald weitere nach.

Mit freundlichen Grüssen

Für die Schriftleitung

Dr. P. Elzer

Erfahrungen mit PEARL in Firmenschulung und Hochschulvorlesungen

H. Mittendorf

Zusammenfassung

In Karlsruhe wird PEARL im Rahmen des Themas der Tagung an zwei Stellen behandelt:

a) in der Schule für Prozeßrechner der Siemens AG, E 36

b) als Teil meiner Vorlesung "Prozeßprogrammiersprachen und Anwenderprogrammsysteme" an der Universität Karlsruhe.

In a) steht das Kennenlernen von PEARL und der Siemens-Software-Umgebung im Vordergrund, in b) wird PEARL als ein (wenn auch das wichtigste) Instrument der Prozeßprogrammierung behandelt. Außer den Grundlagen der Sprache werden in b) zwei große Programmsysteme als Beispiele für PEARL-Programmiertechnik von Anwenderproblemen behandelt. Zusätzlich wird im zweiten Beispiel im einzelnen auch auf das Bedienkonzept und eine gezielte Optimierung zur Verbesserung der Alarmauflösungszeit eingegangen.

Abstract

PEARL is treated - as special issue of this workshop - in Karlsruhe at two occasions:

a) in the "Schule für Prozeßrechner" of the Siemens AG, E 36 and

b) as a part of my lecture "Prozeßprogrammiersprachen und Anwenderprogrammsysteme" at the university of Karlsruhe.

At a) there will be the main subject the knowledge of PEARL and of the environment of the Siemens system software, in b) is PEARL not the main topic of the lecture but the most useful tool to implement real time (process programming) software. Additionally to the fundamentals of PEARL there will be treated in b) two big user systems as examples of the PEARL programming technique. In detail there will be demonstrated at the second example a special user dialog technique and the optimization of the efficiency of interrupt processing.

0 Vorbemerkungen

PEARL wird bei Siemens (genauer im Unternehmensbereich E am Standort Karlsruhe) in zwei Veranstaltungen dargeboten:

a) an der "Schule für Prozeßrechnertechnik", E 36 im Werk Khe

b) an der Universität Karlsruhe am Lehrstuhl für Informatik III

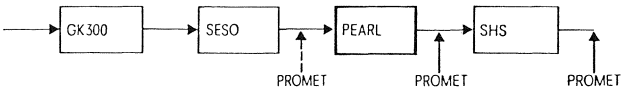
Zu a) sehen Sie ein Kursblatt (Seite 67 der Zusammenfassung aller Kursbeschreibungen) mit den wesentlichen Voraussetzungen zu diesem Kurs. PEARL ist also dort eingeordnet in ein allgemeines Software-Schulungskonzept, worin außer PEARL für die Siemens Systeme 16 Bit auch die Software-Umgebung behandelt wird, wobei gewisse Voraussetzungen über den notwendigen Kenntnis-Umfang gemacht werden. Dazu das Bild 1.

Zu b) sehen Sie in den nächsten beiden Bildern (2,3) die Einordnung von PEARL in meine allgemeine Vorlesung "Anwendersysteme und Prozeß-Programmiersprachen". Ich halte dabei 2 Doppelstunden für die Einführung der Grundbegriffe für ausreichend. Ich gehe dann (Bild 3) erst später (d.h. nach der Behandlung von weiteren Prozeß-Programmiersprachen und typischer Anwendungsfälle) noch einmal anhand von aus der Praxis genommenen Beispielen auf einzelne Probleme der Programmierung mit PEARL ein. Die Vorlesung wird dann beendet mit einer Exkursion in eines der Rechenzentren der Siemens AG am Standort Karlsruhe.

In den nächsten beiden Punkten werden nun die jeweiligen Gesichtspunkte zu a) und b) vertieft. Dabei werden in b) gezielt die m.E. wesentlichen didaktischen Prinzipien des Entwurfes von Prozeß-Anwendersystemen herausgearbeitet, die beim Umgang mit PEARL zu beachten sind:

PEARL

PEARL (Process and Experiment Automation Realtime Language)



Prerequisites:

- Attendance of the SESO course or equivalent knowledge

Course objectives:

- The participant should be able to formulate PEARL modules in basic PEARL on his own, translate them with the PEARL compiler PC30 and link the individually generated object code elements with the BD30 linkage editor.
- He should be able to load the linked objects, test them with the PEARL diagnostic system using TEPOS and execute them.

Summary of contents:

- Basics of programming
- Significance of PEARL in connection with other programming languages
- PEARL
  Character set
  Basic structure of a module
- PROBLEM part
  Data types (FIXED, FLOAT, BIT, CHAR, CLOCK, DURATION)
  Records (fields, structures)
  Declaration, specification
  Task, procedure
  Arithmetic expressions
  Statements for controlling sequential program flow (GOTO, IF, CASE, ON, loop)
  Tasking statements (ACTIVATE, TERMINATE, etc.)
  Input/output transfers (file transfers, transfers with the process peripherals)
  Alarm processing
  Coordination counter (semaphore)
- SYSTEM part
  Hardware structure and terminology of the 300/16-Bit Systems (process peripherals, standard peripherals)
  Signals
  Interrupts
- Programming exercises
- Handling the compiler
  Linkage editing
  Loading
  Testing
- Assembler connection

Duration: 10 days

Material: The following material will be distributed at the course:

- PEARL course folder
- PC 30/PEARL 300 manual

Remarks:

- The course will also be held in English upon request.

B i l d 1: Inhalt des PEARL-Kurses
(mit frdl. Genehmigung der Prozeßrechner-Schule der E 36 der Siemens AG)

a) die Erarbeitung und Implementierung des jeweiligen Bedienkonzeptes,

b) die Minimierung der Antwortzeiten alarmgesteuerter Prozesse.

1 Gesichtspunkte der Prozeßrechnerschule

Hier erscheinen 2 Gesichtspunkte wesentlich:

a) Der PEARL-Anwender hat außer der intimen Kenntnis von PEARL die Systemumgebung zu kennen (dazu im Bild 4 eine graphische Darstellung dieser Umgebung)

b) Die speziell in der Siemens-Implementierung benutzbaren Systemgeräte (im Sinne von PEARL) und die im Laufzeitsystem implementierten SIGNALs. Für Einzelheiten muß auf unser Manual zum PC30 (Seiten 86 ff. für Systemgeräte und 177 ff. für SIGNALs) verwiesen werden (siehe /1/).

Teilnehmer eines Kurses der Prozeßrechnerschule sind in aller Regel potentielle Anwender der Siemens-Rechner. Daher steht das Umgehen mit dem Rechner und das Arbeiten am Rechner im Vordergrund.

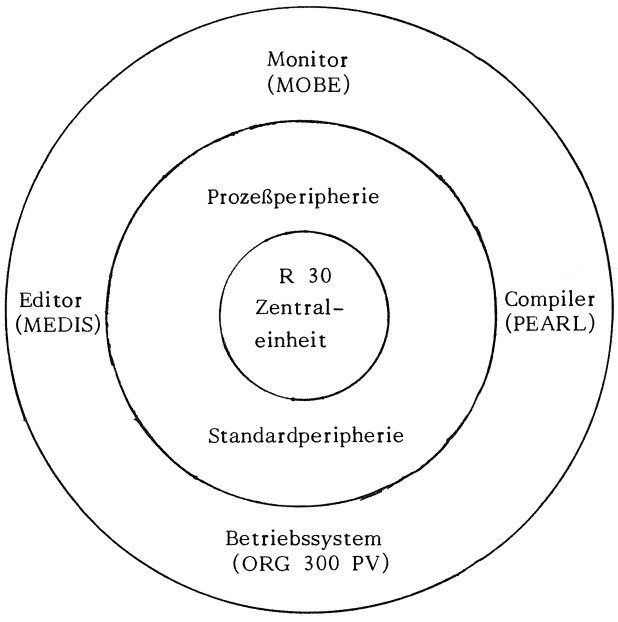
Nr.	T h e m a	Dat.
		1981
1	Grundlegende Gesichtspunkte der Automatisierung	21.10.
2	ALGOL und FORTRAN für Prozeß-anwendungen	28.10.
3	PEARL: Grundlagen, Systemteil und Prinzipien der Ein/Ausgabe	04.11.
4	PEARL: Sprachmittel zur Programmierung von Algorithmen und zur Organisation von Parallelarbeit	11.11.
5	Prozeß-FORTRAN: Ergänzungen zum Tasking und Anwendungsbeispiele	25.11.
6	Prozeßprogrammsysteme in der Analysentechnik	02.12.
7	Prozeßprogrammsysteme in der Gasfernversorgung	09.12.

B i l d 2: 1. Teil der Vorlesung WS 81/82

Nr.	T h e m a	Dat.
		1981
8	Regressionsrechnung und lineare Optimierung als Mittel der Modell-bildung beim Prozeßrechnereinsatz	16.12.
		1982
9	PEARL: Programmierung eines Systems zur zyklischen Datenerfassung	13.01.
10	Programmsysteme zur Bearbeitung von Grundaufgaben der Prozeß-programmierung (SIMAT) PEARL: Programmierung eines Meldedruckers	03.02.
11	Teil 1: Aufgabenstellung und Systemteil	10.02.
12	Teil 2: Die Task- und UP-Struktur	17.02.
13	Exkursion ins RZ Khe, der Siemens AG	24.02.

B i l d 3: 2. Teil der Vorlesung WS 81/82

FLOW of PEARL-Training



. System division
- Hardware structure (peripherals)
- Short introduction into operating system

. Significance of PEARL in connection with other programming languages

. Handling
- Editing (Introduction into MEDIS)
- Linkage
- Assembler connection

. Programming exercises

B i l d 4: Software-Umgebung von PEARL

2 Gesichtspunkte meiner Vorlesung

Hier wird kein allgemeiner oder auf eine bestimmte Anlage bezogener PEARL-Kurs geboten, sondern PEARL als Mittel zur Formulierung von prozeßspezifischen Aufgabenstellungen gesehen. Obwohl PEARL zweifelsfrei das beste bekannte Mittel hierzu ist, wird auch Prozeß-FORTRAN in 2 Doppelstunden angeboten (Hinweis: Siemens hat neben 40 PEARL-Kompiliersystemen über 400 Prozeß-FORTRAN-Compiler mit zugehöriger Prozeß-Library geliefert).

Wesentlich auch für den Studenten: PEARL ist zwar wichtigste aber nicht alleinige höhere Sprache zur Prozeß-Programmierung. Hinweis: Es wird auch ein mittleres Beispiel in Prozeß-FORTRAN vorgetragen. Wir müssen auch der Tatsache ins Auge sehen, daß die Beliebtheit von FORTRAN noch immer nicht im Abnehmen begriffen ist.

2.1 Grundlagen von PEARL

In jeweils einer Doppelstunde werden die Grundlagen des Systemteils und der Ein-Ausgabe einerseits und der algorithmische Teil mit Tasking andererseits vorgetragen. Dabei finden keine Übungen statt, da dies den Rahmen der Gesamt-Vorlesung sprengen würde.

Selbstverständlich werden die Sprachkonstrukte in kleinen Beispielen erläutert; aber über den Programmentwurf im ganzen wird hier noch nichts ausgesagt. Dies erfolgt erst in den beiden später behandelten Mehr-Task-Beispielen.

2.2 Ein 4-Task-Beispiel mit zyklischer Datenerfassung und Ausgabe

Es ist m.E. wichtiger, dem Studenten die Prinzipien des Programm-Entwurfs an zunächst kleineren Beispielen zu verdeutlichen, als "im Trockenkurs" eine erschöpfende Darstellung aller PEARL-Sprachelemente zu geben. Daher wird an einem 4-Task-Beispiel mit "schwacher Prozedurierung" zunächst das Prinzip der Behandlung zyklischer Prozesse dargelegt.

Der Entwerfer sollte sich beim Entwurf zunächst folgende Fragen stellen und versuchen, Antworten systematisch zu entwickeln:

a) Wieviel von einander unabhängig arbeitende Geräte muß das System bearbeiten?

b) Wie erfolgt die Koordinierung zwischen den Prozessen?

c) Welche Teilaufgaben lassen sich definieren und wie sind sie in Prozeduren umsetzbar?

d) Braucht das System einen besonderen Bediener-Prozeß?

e) Wie erfolgt die "Grundversorgung" der globalen Daten?

f) Wie sollen Laufzeitbesonderheiten abgefangen werden?

g) Wie erfolgt der System-Anlauf?

h) Wie erfolgt das "definierte Ausschalten" des Systems?

Sind diese Fragen beantwortet, so läßt sich die Anzahl der erforderlichen Tasks wie folgt festlegen:

a) Jedes unabhängig arbeitende Gerät wird durch eine Task bearbeitet.

b) Dabei erfordert auch das Bediengerät (oder der Bedienmodus eines Gerätes) eine eigene Task.

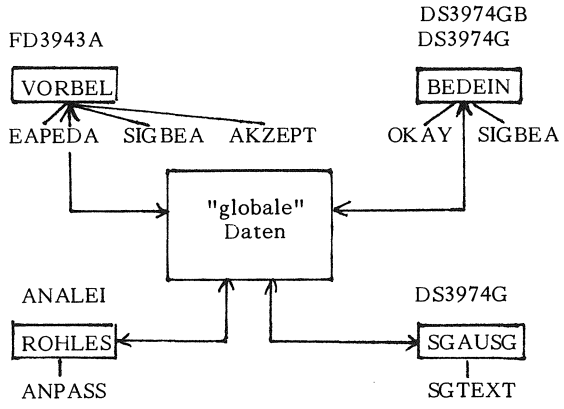
c) Eine Task ist für den Start und evtl. die Grundversorgung der globalen Daten erforderlich.

d) Eine Task sollte für den System-Stop vorgesehen werden (dies kann entfallen, wenn ein allgemeines Wiederanlaufsystem vorhanden ist, das mit einem vorher erstellten "HSP-Abbild" arbeitet).

Grundregel: Im Zweifelsfall lieber eine Task zuviel, als eine zuwenig vorsehen.

Im Bild 5 sehen Sie das Schema für einen solchen Grobentwurf mit schwacher Prozedurierung.

Hier sind die Task's jeweils eingerahmt dargestellt; benötigte Geräte sind darüber mit ihrem Siemens--Systemnamen angebracht. Unter den Task-Namen stehende Namen bezeichnen die benötigten Prozeduren. Die globalen Daten (genauer: Daten auf Modulebene) sind in der Mitte angeordnet.



B i l d 5: Das 4-Task-Beispiel (mit schwacher Prozedurierung)

Für die Festlegung der obigen Task-Struktur wird dem Studenten das folgende Argumentarium zur Verfügung gestellt:

a) Die Zerlegung eines Programmsystems erfordert nach obiger "Standardregel" n+2 Tasks, wobei n die Anzahl der unabhängig betriebenen Geräte (bzw. Betriebsmodi, wenn ein Gerät in mehreren solcher Betriebsmodi benutzt wird) ist. Zu den "Geräte"-Tasks tritt dann noch jeweils eine Start- und eine Stop-Task.

b) Nach dieser Regel sind erforderlich:
eine Task (VORBEL) zur Vorbelegung (Floppy) der Grunddaten
eine Task (ROHLES) zum Lesen der "Rohdaten" über Analogeingabe
eine Task (SGAUSG) zur allgemeinen Ausgabe über das Sichtgerät
eine Task (BEDEIN) zur Bedieneingabe über Bediengerät.

c) Die "Start-Task" ist in die Task BEDEIN verlagert; eine "Stop-Task" ist nicht erforderlich, da Start-/-Stop-Betrieb durch einen System-Wiederanlauf erreicht wird.

d) Die Geräte-Funktionen der Floppy/Datei einerseits und Sichtgerät/Drucker andererseits laufen nicht simultan zueinander; sie sind also in jeweils nur einer (!) Task (hier VORBEL bzw. SGAUSG) zu bearbeiten.

e) Das Sichtgerät 3974 ist hier für die Funktionen "zyklisch umlaufende" und "spezifische" Ausgabe einerseits und (!) "Bedieneingabe" andererseits als "2-Modus-Gerät" betrieben. Daher dient hier das physikalische Gerät "3974" mit 80 Zeichen pro Zeile und 24 Zeilen pro Seite sowohl als Systemgerät "DS3974GB" (Anwendername BEDIEN) für BEDEI als auch als Systemgerät "DS3974G" (Anwendername BEDAUS) für SGAUSG.

   Diese beiden "Geräte" sind echt simultan betreibbar. Allerdings geben die simultane Benutzung der Bedienung und der zyklischen Ausgabe Anwender-Hantierungs-Probleme (STX-Taste benutzen, "schnell" bedienen!). Zweckmäßig wäre ein zweites Sichtgerät.

Resultat: Es sind genau 4 Tasks erforderlich.

2.3 Ein 8- (bzw. 9-)Task-Beispiel mit alarmgesteuerter Erfassung von Binärsignalen, Ausgabe über SG und Drucker, sowie Textbedienung

Im 2. Teil der Vorlesung wird dann ein großes Anwenderbeispiel behandelt (siehe auch Kapitel 7 von /2/). Dabei steht zunächst die Erarbeitung der Task-Struktur im Vordergrund.

Zu Beginn sei die Aufgabenstellung kurz geschildert: "Meldungs- und Tastatur-Alarme von Prozessen" sollen in einem Programmsystem (Abkürzung: METAP) in geeigneter Weise bearbeitet werden. Dabei ist schnellstmögliche Erfassung der durch Binär-Pegel dargestellten Anlagen- und Bedien-Zustände sicherzustellen. Die Meldungstexte sind simultan und in der zeitlichen Reihenfolge des Eintreffens auf Sichtgerät und schreibendem Gerät zur Ausgabe zu bringen. Dabei ist außer Datum und Uhrzeit eine Meldung zu kennzeichnen durch Meldungs-Nummer, sog. "Klartext" (bestehend aus technischen Kurzbezeichnungen) und Änderungshinweis (entsprechend dem Wert des Binärpegels). Entsprechend der Stellung der Bedienungstasten, die simultan zur Meldungserfassung bearbeitet werden muß, sind bestimmte Zustände der Ausgabe auf dem Sichtgerät bzw. Zusatzbezeichnungen im Text zu realisieren.

Vergleicht man mit dem vorherigen 4-Task-Beispiel, so kann man grob folgendes feststellen:

a) Da die Erfassung alarmgesteuert erfolgt, ist das vorherige Gerät "Analogeingabe" logisch durch "dynamische Digitaleingabe" (verbunden mit geeigneten Geräten zur statischen Digitaleingabe) zu ersetzen. Daher ist hierfür keine weitere Task erforderlich!

b) Systemstart ist explizit erforderlich, da die Bedieneingabe über Tasten als von obiger getrennte weitere alarmgesteuerte Erfassungs-Task vorzusehen ist (Tastenbedienung muß simultan erfolgen können). Im vorherigen Beispiel wurde Systemstart über VORBEL mit-erledigt.

c) Da auch eine Arbeitsweise ohne automatischen Wiederanlauf möglich sein muß, ist zusätzlich eine System-Stop-Task erforderlich.

d) Zusätzlich ist simultane Ausgabe auf schreibendem Gerät gefordert.

c) und d) liefern neue Tasks gegenüber dem vorherigen Beispiel.

Man kann aufgrund der Aufgabenstellung und der dazu erforderlichen Geräte die Zerlegung in Tasks auch direkt angeben. Man stellt dann fest, daß folgende Programme erforderlich sind:

a) zur Analyse der Meldungs-Alarme eine Task MELDAL,

b) zur Analyse der Bedienungs-Alarme eine Task BEDIAL,

c) zur Bearbeitung der Ausgabe auf dem Sichtgerät eine Task BEARSG,

d) zur Bearbeitung der Ausgabe auf dem Blattschreiber eine Task BEARBS, (für Blattschreiber wird später ein Drucker eingesetzt),

e) zum Start des gesamten "main"-Programms eine Task MAINST,

f) zum Stop des gesamten Programm-Systems eine Task SYSTOP.

Der Grundaufwand besteht also aus 6 Tasks.

2.3.1 Zusatzinformation bei kritischen Anlagenzuständen

Gelegentlich könnten Störmeldungen kritische Anlagenzustände anzeigen (man denke z.B. an Störungen in Kernkraftwerken). Dann wäre der Wunsch verständlich, bei

vorgebbaren (durch bestimmte Kombination von Störmeldungen gekennzeichneten) Anlagenzuständen auf Hintergrundspeicher ein Zustandsbild der Gesamt-Anlage zu hinterlegen und für spätere Analyse des Störfalls bereitzuhalten.

Die Lösung einer solchen Zusatzaufgabe ist

a) durch eine besondere Task zu realisieren, da in der zeitkritischen Erfassung (MELDAL oder BEDIAL) selbst eine derartige Aufgabe nicht zu realisieren ist (sie würde die Alarmauflösungszeit in unerträglicher Weise erhöhen),

b) in dem Teil, der mit der Meldungsanalyse erfolgt, so zu gestalten, daß diese zeitlich nur wenig zusätzlich belastet wird.

Diese Zusatztask (AUSLES) wird also nur Information auslesen, die im Meldungs-Analyseteil zusätzlich HRP-resident ausgegeben wird, wenn die ermittelte Meldungsnummer eine "kritische" ist.

Es ist dann zu überlegen, wie diese Zusatzinformation auszusehen hat. Die sicherste Methode ist, bei jeder kritischen Meldungsnummer den gesamten Anlagenzustand als Wechselpuffer-Element zu hinterlegen und ggfs. (wenn nämlich der eine der 2 Wechselpuffer gefüllt ist), den zugehörigen entsprechend vorbesetzten Semaphor "zu erniedrigen". Durch diese Strategie wird nur mäßig Haupt speicherplatz benötigt, jedoch bei länger andauernden Störfällen sehr viel Information auf Externspeicher hinterlegt. Hier wird nicht über Strategien zum Abarbeiten dieser Information diskutiert. Sinn dieser Zusatzaufgabe ist es, den Interessenten von PEARL (bzw. dem Studenten, der die Möglichkeiten von PEARL kennenlernen soll), an einem Beispiel klarzumachen, wie man eine solche Zusatzaufgabe sinnvoll "angeht".

Als Ergebnis dieser Überlegungen ist festzuhalten:

a) Erster Teil der Zusatzaufgabe ist die Erweiterung der Meldungs-Analyse um die ggfs. nötige Ablage eines Pufferelementes mit entsprechender Freigabe der Auslesetask.

b) Zweiter Teil der Zusatzaufgabe besteht in der Formulierung dieser Auslesetask, die als Gerät einen Externspeicher benötigt.

c) Ggfs. ist kritisch zu prüfen, ob die Meldungsanalyse im Fall kritischen Anlagenzustandes nicht unzulässig verzögert wird.
(Davon wird im folgenden noch die Rede sein).

2.3.2 Hinzufügung eines Bedienprozesses

Für den realen Bereich des oben beschriebenen Systems METAP muß es möglich sein, die Meldungstexte (Klartexte mit Änderungshinweisen) auch nach Inbetriebnahme der Anlage in das System einzubringen. Die Textdaten müssen also "bedienbar" sein.

Dieser Zusatzwunsch wird durch die Task DATBED verwirklicht. Da DATBED sehr groß ist, wird den Studenten nur andeutungsweise vermittelt, wie ein solches Bedienprogramm aufzubauen ist; dazu folgende Hinweise:

a) Es ist in DATBED zweckmäßiger Weise sowohl die erstmalige Datengenerierung für alle Texte, als auch die nachträgliche Hinzufügung bzw. Modifizierung vorhandener Texte zu realisieren.

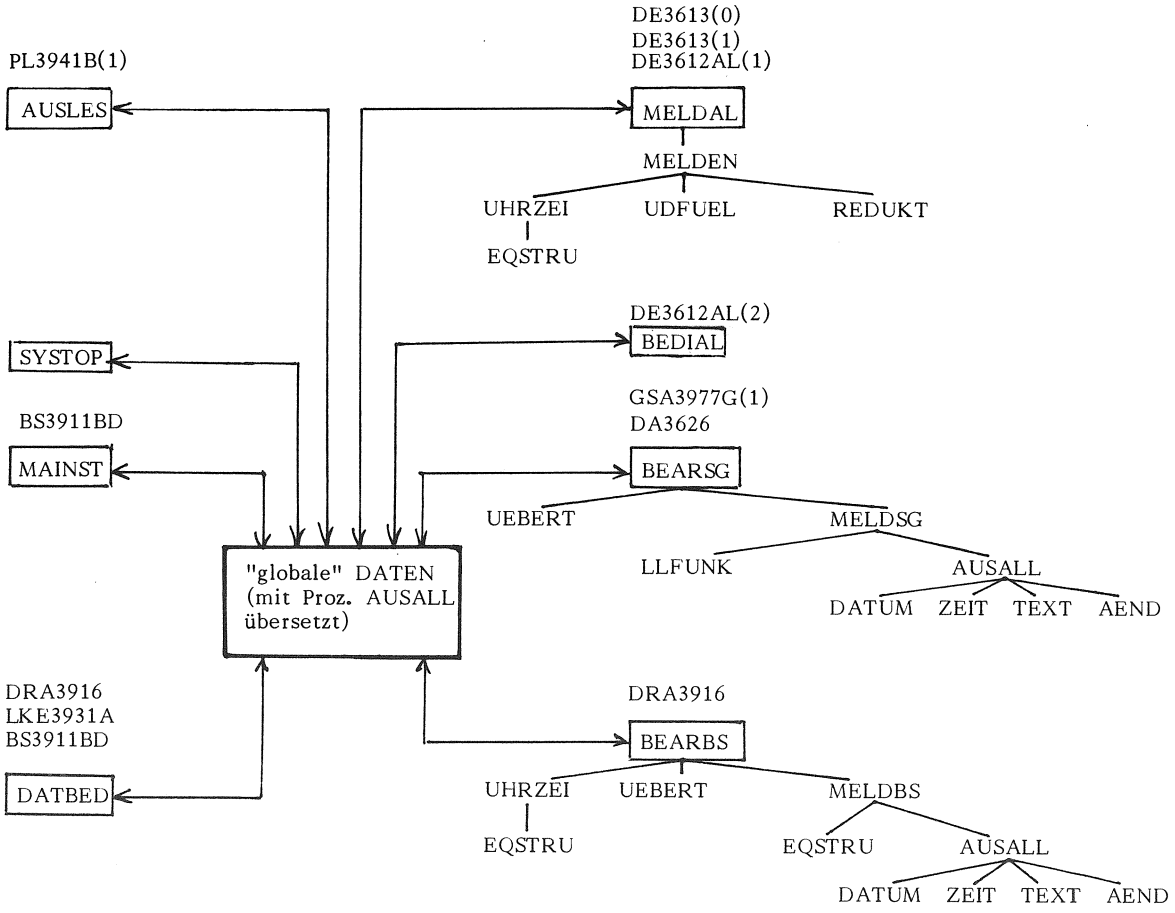
b) Als Eingabegerät für DATBED ist daher sowohl eine Lochkarteneingabe (später durch Floppy Disk ersetzt), als auch eine "Bedieneingabe" über das Bediensichtgerät (bzw. den Bedienblattschreiber) vorzusehen.

c) Aus den Texten werden Textstücklisten und zugehörige Indexlisten ermittelt. Daher ist auch ein Drucker vorzusehen, auf dem diese Information auf Wunsch ausgebbar ist. Der dort erscheinende Drucktext gibt alle Listen als Initialisierungs-Anweisungen in PEARL heraus. Das bedeutet, daß der Output auch direkt als PEARL-Quellcode verwendet werden könnte; damit wäre eine effiziente Hauptspeicherlösung des gesamten Systems in PEARL gewinnbar. Allerdings ist im Test diese Lösung nie wirklich zur Ausführung gekommen.

d) Bei der Modifizierung von Texten ist zu beachten, daß zwar in der Regel nur lesend (beim Textvergleich) zu den Listen zugegriffen wird und die einzigen schreibenden Zugriffe im Eintrag der neuen Textinformation bestehen, daß aber aus Gründen einer "sauberen" Programmierung beim Listenzugriff über zugehörige Semaphore zu arbeiten ist. Allerdings ist dies nur bei echter Online-Bedienung notwendig (läßt man es auch dort weg, so werden ggfs. Texte unmittelbar nach der Textbedienung "verstümmelt" ausgegeben).

e) Zur Erleichterung der Textbedienung ist eine geeignete Syntax der Bedieneingabe zu überlegen und dem Bediener im Dialog zu verdeutlichen. Hierbei ist jedoch der Aufwand nicht zu groß zu wählen.

Es ist natürlich nicht möglich, ein so großes Programm im Rahmen dieser Vorlesung weiter im einzelnen zu behandeln, jedoch darf der Hinweis nicht vergessen



B i l d 6: Das 8-Task-Beispiel (mit starker Prozedurierung)

werden, daß auch über die nötigen SIGNALs des Bediengerätes nachgedacht werden muß. Hier ist allerdings keine absolute Sicherheit möglich. Immerhin ist im Laufe zahlreicher Versuche bisher kein unvorhergesehener Programmzustand aufgetreten.

Es würde den Rahmen dieses Berichtes sprengen, wenn auch noch über die Prozedurierung von METAP gesprochen würde. Es sei hierzu auf das folgende Bild 6 zu dem 8-Task-Beispiel verwiesen.

2.3.3 Optimierung zur Verbesserung der Alarmauflösungszeit

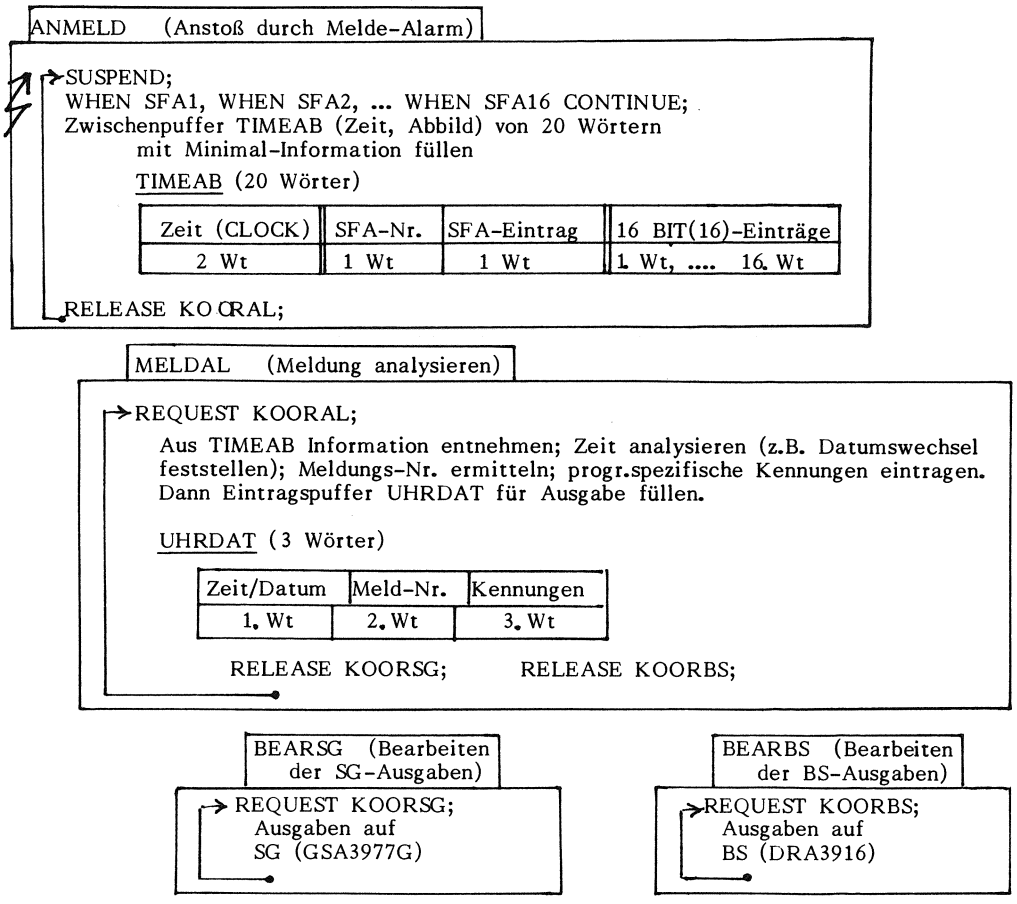
Das in Assembler geschriebene System hat eine Auflösungszeit von 2 ms. Dazu wurde allerdings ein vereinfachtes Organisationsprogramm benutzt.

Messungen an dem in PEARL geschriebenen System ergaben eine Auflösungszeit von 21 ms (/3/). Diese um den Faktor 10 schlechtere Lösung gibt Anlaß zu der Überlegung, ob nicht eine grundsätzlich veränderte Programmstruktur gesucht werden muß. Tatsächlich bringen Verbesserungen in den Unterprogrammen (z.B. inline-code-Formulierungen oder Modifizierungen der Datums-/Uhrzeitermittlung) nur unwesentliche Verbesserungen.

Es wird daher nun den Studenten nahegebracht, wie vom Prinzip her eine Verbesserung dadurch erreicht werden kann, daß eine weitere Task formuliert wird. Diese hat die Aufgabe, im Falle eines Meldungsalarms nur nach Art eines "schnellen Abgriffs" eine Minimalinformation über die Anlage auszulesen, dann sofort als Element in einen Umlaufpuffer einzutragen und danach die (jetzt modifizierte) Task MELDAL freizugeben (in ASS-Denkweise: "über Koordinierungszähler zu starten")

Man hätte hierzu also eine weitere Task ANMELD (Anstoß für MELDAL) zu formulieren und eine geeignete Struktur für den Zwischenpuffer zu überlegen. Im folgenden Bild 7 sehen Sie, wie nach Einführung dieser Zwischenstruktur TIMEAB (Zeit und Abbild) das gegenseitige Verhältnis der 4 Tasks ANMELD, MELDAL, BEARSG und BEARBS sich nunmehr darstellt.

Es ist wichtig, daß der Student hier "mit nach Hause" nimmt, daß eine sinnvoll geplante Programm- bzw.



B i l d 7: Beziehungen zwischen den 4 Tasks ANMELD, MELDAL, BEARSG, BEARBS nach der Optimierung des Systems

Datenstruktur oft ein "langsames" in PEARL geschriebenes System überraschend einfach "schnell" macht. Die ersten Versuche mit diesem optimalen System haben eine Alarmauflösungszeit von etwa 4-5 ms erbracht (gegenüber 21 ms vorher).

2.4 Exkursion

Zum Schluß soll nicht unerwähnt bleiben, daß zur Motivierung des Umgangs mit PEARL natürlich auch das Arbeiten mit einem realen System gehört. Da meine Vorlesung kein PEARL-Kurs ist, aber doch zum Umgang mit PEARL das Arbeiten an einer realen Anlage gehört, wird jeweils in der letzten Doppelstunde eine Exkursion zum RZ Karlsruhe durchgeführt. Hier kann sich der interessierte Student "vor Ort" von der Funktionsweise des oben behandelten 4- bzw. 8-Task-Beispiels persönlich überzeugen. Die dazu nötige Rechenzeit stellt dankenswerter Weise das Haus Siemens zur Verfügung.

/1/: PC30/PEARL 300; Manual zum PEARL-Compiler der Siemens AG,
Bestell-Nr. P71100-D3010-X-X-35 vom 30.07.80

/2/: Kappatsch, A., Rieder, P. und H. Mittendorf: PEARL, Systematische Darstellung für den Anwender (300 Seiten, Oldenbourg 1979)
ISBN 3-486-23021-2

/3/: Lorenz, Udo: Zeitmessung des Ablaufs von Prozeßprogrammen. Diplomarbeit SS79, Fachhochschule Khe, Fachbereich N

Dr. rer.-nat. Mittendorf, Horst
in Firma Siemens AG, E STE 33
7500 Karlsruhe, Rheinbrückenstraße 50
Tel.: (0721)-595-2284

Erfahrungen mit PEARL-Schulungen im industriellen Bereich

Dr. Hans Windauer, Lüneburg

Zusammenfassung

Dieser Kurzbericht faßt die Erfahrungen aus acht PEARL-Seminaren und -Schulungskursen im industriellen Bereich zusammen.

Schlüsselwörter: PEARL Schulungskurs

Summary

This short report summarizes experiences gained in eight PEARL seminars with participants from industrial institutions.

Key words: PEARL seminar

1. Grundlage der Erfahrungen

Dieser kurze Bericht fußt auf fünf dreitägigen, einem zweitägigen und zwei eintägigen Schulungskursen, die der Autor in den Jahren 1976 bis 1979 durchgeführt hat.

Die Teilnehmer kamen aus den Bereichen

- Chemische Industrie
- Förder- und Lagertechnik
- Hersteller von DV-Anlagen
- Kernforschung
- Luft- und Raumfahrt
- Maschinenbau
- Militär
- Rundfunktechnik
- Stahlindustrie (Hüttenwesen)

Die meisten kannten Assembler oder Echtzeit-FORTRAN; weniger Teilnehmer kannten ALGOL 60 oder PASCAL. Kenntnisse in Prozeßprogrammierung waren immer vorausgesetzt.

PEARL war damals noch nicht so verbreitet und gefestigt wie heute, d.h. die Teilnehmer waren in der Regel zwar neugierig, aber auch kritisch bis skeptisch. Deshalb erforderten die Kurse immer auch viel Motivationsarbeit für PEARL.

2. Designziele von PEARL

Die Haltung vieler Teilnehmer war durch folgende Feststellungen gekennzeichnet:

> "Nun programmieren wir schon zu 80% in FORTRAN anstatt in Assembler. Die restlichen 20% lohnen die Einführung von PEARL nicht."

Denn:

> "Die PEARL-Anweisungen für E/A und Tasking sind doch nur syntaktischer Zucker. Beim Übergang von einem Rechnersystem zu einem anderen können die wenigen neuen Betriebssystem-Aufrufe oder RT-FORTRAN-Calls für E/A und Tasking schnell gelernt werden."

Offenbar wurden zu oft die Tasking- und E/A-Eigenschaften von PEARL, nicht jedoch seine Vorzüge im algorithmischen Bereich betont. Außerdem war der eigentliche Vorteil der Integration von E/A und Tasking, die Definition einer Schnittstelle zu Echtzeitbetriebssystemen in seinen Auswirkungen zu oft nicht bewußt.

Deshalb sollten zu Beginn einer PEARL-Schulung alle Designziele und alle wesentlichen Merkmale von PEARL überzeugend, d.h. anhand praxis-naher Beispiele erläutert werden:

Designziele von PEARL [1]:

- Flexibilität der Programme
- Selbstdokumentation der Programme
- Einfache Handhabbarkeit der Sprache
- Einfaches Erlernen der Sprache.

Die wesentlichen Merkmale von PEARL [1]:

- Umfangreiche Menge von Task-Anweisungen und Mechanismen zur Behandlung von Zeitereignissen und Einplanungen
- Synchronisation mittels Sema- und Boltvariablen
- Systemteil
- Übliche Operationen für die Behandlung von Files
- Strukturen, Bit- und Zeichenketten
- Referenzen für indirekte Adressierung
- Benutzerdefinierte Datentypen (TYPE) und benutzerdefinierte Operatoren
- Getrennte Compilierung von Moduln
- Definition eines virtuellen Echtzeit-Betriebssystems.

3. Tragfähigkeit von PEARL

Wichtig ist, darauf hinzuweisen, welche Institutionen hinter PEARL stehen, damit die Kursteilnehmer die Bedeutung und Tragfähigkeit von PEARL erkennen.

Erwähnt werden sollten zumindest die beteiligten Hersteller, der VDI/VDE, der BMFT mit seinem Fördervolumen, die Großforschungseinrichtungen sowie nun natürlich die Normung durch den DIN.

Heute kann auch schon von einigen realisierten Anwendungsprojekten berichtet werden, was sicherlich am besten überzeugt.

Bei allen durchgeführten Kursen kamen die Teilnehmer auf die Konkurrenten von PEARL zu sprechen: IRTF (Industrial Real Time FORTRAN) oder FORTRAN 75 (des VDI/VDE), CORAL 66, RTL, (Concurrent-) PASCAL und neuerdings natürlich Ada. Der Kursleiter sollte PEARL und diese Sprachen einordnen können und grob über ihren Entwicklungsstand und ihre Normungssituation Auskunft geben können.

4. Berücksichtigung verschiedener Zielgruppen

Bei den durchgeführten Kursen gab es im wesentlichen drei Gruppen: Teilnehmer, die bisher nur mit Assembler programmiert hatten, Teilnehmer, die bisher nur mit FORTRAN programmiert hatten und Teilnehmer, die sowohl Assembler als auch FORTRAN kannten.

Allen drei Gruppen neu waren die Hilfen, die PEARL für ein strukturiertes Programmieren bietet. Diese Sprachelemente mußten dementsprechend betont werden (siehe auch Punkt 5).

Die Assembler-Gruppe war immer an Effizienzfragen interessiert. Typische Fragen waren: "Was macht der Compiler daraus?", "Wie ist das Zusammenspiel zwischen Sprache, Compiler, Laufzeitsystem und dem Betriebssystem mit seinen E/A-Werken und Treibern?"

Die FORTRAN-Gruppe hatte die üblichen Schwierigkeiten bezüglich des Arbeitens mit Referenzen und hätte gerne hierauf mehr Zeit und Beispiele verwendet gesehen.

Falls also die Planung eines Kurses es erlaubt, sollten der Aufbau des Kurses, die Betonung von Schwerpunkten und die Auswahl der Beispiele auf die Vorbildung der Teilnehmer abgestimmt werden.

5. Betonung des algorithmischen Teils

Die Vorzüge von PASCAL, insbesondere seine Möglichkeiten zum Arbeiten mit problemorientierten Datentypen, werden überall gepriesen und von vielen akzeptiert, auch wenn sie sie gar nicht selbst kennengelernt haben. PEARL wird als Prozeßsprache eingeordnet mit besonderen Sprachmitteln für Tasking und E/A. Daß PEARL in seinem algorithmischen Teil mit Ausnahme von enumerations, variant records und global generators alle Sprachelemente von PASCAL enthält, mit Bit- und Zeichenketten u.a. sogar darüber hinausgeht, ist kaum bekannt. Umso wichtiger ist die Schulung dieser Eigenschaften von PEARL, d.h. der STRUCT- und TYPE-Elemente, zumal sie sehr viel zur Flexibilität und Selbstdokumentation der Programme beitragen, also auch ihre spätere Wartbarkeit und Modifizierbarkeit sehr stark beeinflussen.

Wichtig ist vor allem auch die Darstellung des Modulkonzepts von PEARL, das den Einsatz von PEARL (im Gegensatz zu PASCAL) bei größeren Projekten überhaupt erst erlaubt.

Bei den durchgeführten 3-tägigen Kursen erwies sich folgende Aufteilung als günstig:

1,5 Tage	:	Algorithmik
0,5 Tag	:	Ein/Ausgabe
0,5 Tag	:	Tasking
0,5 Tag	:	komplexes Anwendungsbeispiel.

6. Bedeutung der Programmiermethodik

Ein erfahrener (und eingefleischter) FORTRAN-Programmierer nahm einmal einen PEARL-Sprachreport zur Hand, der nur aus Syntax- und Semantikregeln bestand. Der FORTRAN-Programmierer wollte PEARL kennenlernen und dazu ein bestehendes FORTRAN-Programm aus dem Hochregallager-Bereich in PEARL neu schreiben. Eine Anleitung lehnte er ab. Heraus kam das gleiche Programm mit PEARL-Schlüsselwörtern statt FORTRAN-Schlüsselwörtern. Modul- und Blockstruktur, eigene Datentypen oder Strukturen waren nicht benutzt worden, obwohl das Programm dadurch kürzer und transparenter geworden wäre.

Deshalb sollte ein PEARL-Schulungskurs für Assembler- oder FORTRAN-Programmierer mit einer Anleitung in Programmiermethodik verbunden sein, bei der zumindest die Grundzüge der strukturierten Programmierung und ihre Berücksichtigung in PEARL-Programmen behandelt werden. Ein dreitägiger Kurs ist dazu natürlich zu kurz.

7. Anwendungsbeispiele

Im Bereich der Prozeßautomatisierung sind überschaubar kleine, aber trotzdem aussagekräftige Beispiele schwierig zu finden. Trotzdem sollte der Aufwand nicht gescheut werden, Beispiele zu erarbeiten, die dem Arbeitsgebiet der Kursteilnehmer entstammen. Anhand solcher Beispiele können die Eigenschaften von PEARL am eindringlichsten und auch am vorteilhaftesten demonstriert werden.

Sehr günstig wirkt sich aus, wenn das PEARL-Programm zu einem von Teilnehmern gewünschten Beispiel "live" während des Kurses entwickelt werden kann. Hierfür sollten jedoch mindestens 4 Stunden zur Verfügung stehen.

8. Literatur

[1] KFK-PDV 1: PEARL. A proposal for a process- and experiment automation realtime language. Gesellschaft für Kernforschung mbH, Karlsruhe. April 1973.

Anschrift des Autors:

Windauer, Hans
Entwicklungsbüro Wulf Werum
Glogauer Straße 2a
2120 Lüneburg
Tel.: 04131 - 53344, - 53066

Anwendererfahrung von PEARL bei Kursen und nachträglichem Einsatz in Projekten

Dipl.-Inform. Erwin Fahr, Friedrichshafen

1. Einleitung

Bei der Dornier System GmbH wurde ein vom BMVg in Auftrag gegebenes PEARL-Software-System entwickelt. Eine Adaption erfolgte bisher an folgenden Zielmaschinen:

- Dornier MUDAS Datenprozessor DP432 (16 Bit uP System)
- Dornier MUDAS Datenprozessor DP433 (16 Bit uP System)
- AEG 80-20
- Intel 8086

Die in diesem Rahmen erwähnten Anwendererfahrungen beziehen sich auf die Entwicklung eines vom BWB in Auftrag gegebenen Ausbildungssimulators für die Besatzung des Fla Rak Pz ROLAND, bestehend aus Kommandant und Richtkanonier. Dabei wurde der Dornier MUDAS Datenprozessor DP433 eingesetzt.

MUDAS-P ist ein modulares, universelles Datenerfassungs- und Steuerungssystem, dessen verschiedene Hardware-Module (Steckkarten) über einen parallelen Bus miteinander verbunden sind. Der Datenprozessor DP433 selbst ist ein MUDAS-P-Modul, der innerhalb eines Modulträgers Platz findet.

Der Maximalausbau des Programmspeichers beträgt 64 k Worte à 16 Bit. Eine Aufteilung in RAM und EPROM ist in jeweils 8 k-Schritten möglich. Dem Benutzer stehen 16 Arbeitsregister zu je 16 Bit zur Verfügung.

2. Implementierter Sprachumfang

Da bei der Dornier System GmbH Mikroprozessoren vorwiegend als Bordrechner in Fahrzeugen, Flugzeugen und Satelliten eingesetzt werden, ergaben sich daraus für die Implementierung des PEARL SW-Systems gewisse Randbedingungen. Diese führten dazu, Sprachelemente aus Basis-PEARL nicht zu verwenden, bei denen Effizienzprobleme oder unnötiger Overhead zu erwarten sind:

- Filehandlung
- Formatierung
- Absolutzeit
- Strukturen.

Problembedingt waren darüber hinaus jedoch auch Erweiterungen notwendig:

- Hilfsmittel zur Programmierung verteilter Systeme
- Anschluß externer Routinen.

3. Schwerpunkte bei der Schulung

Bei durchgeführten PEARL-Kursen in unserem Hause zeigte sich, daß der Erfolg von mehreren Punkten abhängt:

- geeignete Größe der Gruppen
- Einteilung der Gruppen unter Berücksichtigung der mitgebrachten Voraussetzungen (Assemblerprogrammierung mit PDV, Erfahrung in höheren Programmiersprachen ohne PDV)
- Bereitstellung geeigneter Übungsmöglichkeiten.

Die Teilnehmerzahl eines Kurses sollte sich auch unter Berücksichtigung des späteren Übungsbetriebes aus ca. 9 - 15 Personen zusammensetzen. Daraus ergeben sich dann 3 - 5 Übungsgruppen von 3 - 4 Teilnehmern. Der theoretische Teil der Schulung kann von einer Lehrkraft durchgeführt werden, wobei

für den Übungsbetrieb eine zweite hinzugezogen werden sollte.

Für den Übungsbetrieb sind Möglichkeiten zu schaffen, bei denen Testprogramme nicht nur bis zur fehlerfreien Syntax entwickelt und getestet werden können. Zum Austesten der Übungsprogramme stand bei den Kursen in unserem Hause eine Testbox zur Verfügung, die über Analogeingänge, Analogausgänge, digitale Ein- und Ausgänge sowie über aktive Eingänge (Interrupts) verfügt.

Abhängig von der Erfahrung der Kursteilnehmer sind die Schwerpunkte bei der Schulung zu variieren. Assemblerprogrammierer mit Erfahrung in PDV sind mit der Verarbeitung von Prozeßdaten und Interrupts vertraut. Die EQU-Anweisung des MUDAS-Crossassemblers erlaubt eine Hardwarebeschreibung auf einer symbolischen Ebene. Bei dieser Zielgruppe sind somit folgende Schwerpunkte zu setzen:

- Syntax der Hardwarebeschreibung
- Syntax der Sprache
- Taskingverhalten (Aktivierung, Synchronisation, usw.)
- Denken in höheren Programmiersprachen.

Kursteilnehmern, die zuvor ausschließlich mit höheren Programmiersprachen, jedoch ohne PDV, gearbeitet haben, genügt eine kurze Einweisung in die Syntax der Sprache. Dagegen müssen

- die Beschreibung der Hardwarekonfiguration
- die Task-Anweisungen und deren Auswirkungen
- die Synchronisation von Tasks
- die Interruptverarbeitung

sehr ausführlich erklärt werden.
Jedoch ist es gerade im Hinblick auf die Durchführung des praktischen Teils einer Schulung von Vorteil, wenn sich ein Kurs aus Vertretern beider Gruppen zusammensetzt.

4. Akzeptanz der Sprache

Die Akzeptanz von PEARL muß auf mehreren Ebenen betrachtet werden:

- Anwender
- Management
- Auftraggeber.

4.1 Anwender

Bei der Akzeptanz der Sprache im Anwendungsfall ist wieder die von der Schulung her bekannte Unterteilung zu machen. Dabei zeigen die Anwender mit ausschließlicher Erfahrung in höheren Programmiersprachen eine positive Haltung, da

- die Aufgabe problemorientiert formuliert werden kann
- ein Betriebssystem existiert
- auf bestimmte Standardfunktionen zugegriffen werden kann.

Der Assemblerprogrammierer dagegen nimmt eine eher distanzierte Haltung ein, die auf folgenden Punkten basiert:

- für den DP433 existiert ein kleines Betriebssystem, bei dem nur die grundlegenden Task-Anweisungen realisiert sind, was in sehr schnellen Zeiten für Taskwechsel resultiert
- im Laufe der Zeit wurde eine Bibliothek mit oft verwendeten Routinen angelegt
- durch die Makro-Eigenschaften des existierenden Crossassemblers ist eine Bibliothek mit Makros, die Sprachelementen einer höheren Programmiersprache nahekommen, aufgebaut.

4.2 Management

Gerade beim ersten Einsatz von PEARL in einem Projekt treten durch die fehlende Erfahrung in der Kosten- und Zeitkalkulation schwer abzuschätzende Unbekannte auf. Dieser Punkt ist jedoch bei allen nachfolgenden Projekten geklärt.

4.3 Auftraggeber

Die positive Haltung des Auftraggebers zu PEARL zeigt sich daran, daß in neuesten Angebotsaufforderungen eine Realisierung in FORTRAN IV oder PEARL gefordert wird. Eine Realisierungsmöglichkeit in Assembler ist nicht mehr gewünscht.

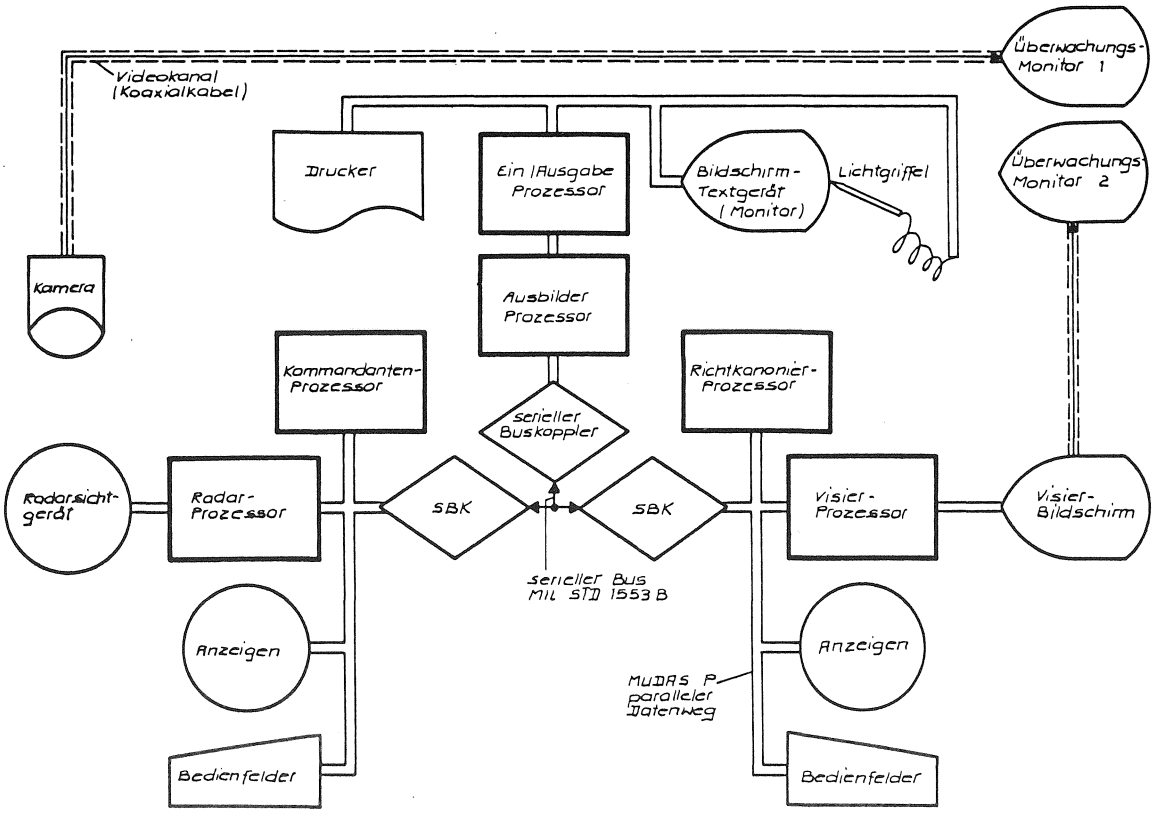


Bild 1: Normalbetrieb

5. Problembeschreibung

Der elektronischen Seite des Ausbildungssimulators liegt ein Mehrprozessorsystem zugrunde, das sich aus der mechanischen Aufteilung und dem analysierten Programmumfang ergibt. Im Gesamtsystem werden 6 Prozessoren eingesetzt, die gleichmäßig auf den Ausbilder-, Kommandanten- und Richtkanonierplatz verteilt sind (A b b. 1). Die Kopplung der 3 Hauptprozessoren erfolgt über einen seriellen Bus (MIL STD 1533 B). Die Treiberroutinen für diesen Buskoppler liegen als anwendergeschriebene PEARL-Programme vor, wohingegen die parallele Kopplung zwischen den beiden Prozessoren in einer Haupt-Baugruppe vom PEARL-Betriebssystem unterstützt wird (verteilte Systeme). Die Aufteilung der einzelnen Aufgaben auf die beiden Prozessoren innerhalb einer Baugruppe erfolgt überall identisch: der Hauptprozessor bearbeitet die Logikfunktionen des Simulators

- Bedienfelder
- Anzeigen
- Daten vom und zum seriellen Bus,

während der zweite Prozessor die einzelnen Bilder aufbaut.

6. Argumente für den Einsatz von PEARL

Für den Einsatz von PEARL im Projekt sprachen folgende Punkte:

- das existierende Betriebssystem, das aufgrund des modularen Aufbaus nicht unnötig viel Speicherplatz in Anspruch nimmt
- Implementierung verteilter Systeme
- Größe der Programme (ca. 20 000 PEARL-Statements in allen 6 Prozessoren)
- Bearbeitung der vielen Endstellen (ca. 150 Eingangssignale
  65 Ausgangssignale
  6 Interrupts (Sammelinterrupts)
- Berechnung vieler komplizierter mathematischer Ausdrücke.

7. PEARL in der Anwendung

Sowohl der Grobentwurf als auch der Feinentwurf der Programme wurden in Anlehnung an PEARL gemacht (Taskanweisungen, CASE-Statement usw.). Die komplizierten mathematischen Ausdrücke, die Simulation des Logikverhaltens und die Treiberroutinen lassen sich in PEARL sehr einfach formu-

lieren. Trotzdem war es nicht möglich, alles in PEARL zu programmieren. Das Gesamtproblem mußte unter mehreren Gesichtspunkten analysiert werden:

- Zeitbedingungen
- Speicherplatzanforderungen
- Codierung spezieller Testroutinen.

Bei sehr zeitkritischen Programmteilen mußten diese als externe Routinen zeitoptimiert in Assembler geschrieben werden. Diese externen Routinen können dann vom PEARL-Programm aus aufgerufen werden.

Das Modulkonzept des PEARL-Software-Systems erlaubt es, das Programm für einen Prozessor in mehrere Module aufzuteilen. Eine PEARL-Prozedur, die in mehreren Modulen aufgerufen wird, muß in jedem sie aufrufenden Modul deklariert werden, da Prozeduren nicht global deklariert werden können (reentrantfähig). Hingegen haben auf externe Routinen mehrere Module Zugriff, auch Module anderer Prozessoren.

8. Testerfahrungen

Durch den zeitlich parallelen Verlauf der Entwicklung der ersten Anwenderprogramme und der letzten Arbeiten am PEARL-Software-System war der Einsatz des Debuggers und der Tracefunktionen nicht möglich. Danach waren die Anwenderprogramme bereits so groß, daß die Testhilfsmittel nicht mehr zugebunden werden konnten. Die Tests mußten daher mit den Möglichkeiten, die unser MUDAS-P-Entwicklungssystem bietet, durchgeführt werden. Die Fehlersuche auf Assemblerebene wurde durch ausführliche Binderprotokolle unterstützt.

An der Software-Entwicklung waren 4 Mitarbeiter beschäftigt. Bei gemeinsamen Tests konnte man sich aufgrund der Programmierung in PEARL in die Programme der anderen Mitarbeiter einarbeiten, diese sogar in deren Abwesenheit testen.

9. Entwicklungszeit

Von den 4 an der Softwareentwicklung beteiligten Mitarbeitern wurde ein Arbeitsaufwand von ca. sieben Mannjahren erbracht. Der Quellcode umfaßt ca. 20 000 PEARL-Statements.

10. Zusammenfassung

Der Einsatz von PEARL erlaubte weitgehend ein problemorientiertes Arbeiten. Durch den selbstdokumentierenden Charakter der Sprache war ein Einarbeiten in die Programme der anderen Mitarbeiter relativ einfach.

Ausbildungsplan für die PEARL-Schulung bei Digital Equipment und erste Erfahrungen

Thomas Hänssgen

1. SCHULUNGSKONZEPT

1.1 ZIELSETZUNG

Wie fuer jeden Computerhersteller besteht fuer DEC die Notwendigkeit und Verpflichtung seine Kunden ueber Arbeitsweise und Funktion seiner Produkte zu unterrichten. Darueber hinaus sehen wir unser Ausbildungsprogramm als zusaetzliche Dienstleistung fuer unsere Kunden an. Diesen Service verstehen wir als Austausch von Know-how mit unsere Kunden.

1.2. AUFBAU DER KURSE

1.2.1 MODULARER AUFBAU

Bedingt durch die Komplexitaet und den Umfang des Stoffes haben wir unsere Lehrplaene modular aufgebaut, d.h. durch sinnvolle Lehreinheiten werden Ueberschneidungen und Wiederholungen vermieden.

1.2.2. VORAUSSETZUNGEN

Da bei diesem modularen Lehrplan ein Kurs an den anderen anschliesst, ist es einerseits fuer den Kunden moeglich individuell abgestimmte Ausbildung zu erhalten, anderseits dafuer von grosser Wichtigkeit den fuer ihn richtigen Einstiegspunkt zu waehlen, da ein Kurs auf den Wissensbereich des vorangegangenen Kurses aufbaut und keine Wiederholungen stattfindet.

1.2.3. LERNZIELE

Um den Kunden bei der Auswahl des fuer ihn richtigen Kurses oder der Kurse zu unterstuetzen, bestehen fuer jeden Lehrgang fest definierte Lernziele. Diese bestimmen den Gegenstand, die Qualitaet und Quantitaet, dienen so der Kommunikation und grenzen den Kurs nach oben und unten ab.

1.2.4. PRAXISORIENTIERUNG

Unsere Software-Kurse sind in einen theoretischen und praktischen Teil gegliedert. Im Schulungszentrum sind insgesamt 25 Rechnersysteme fuer die praktische Ausbildung vorhanden.

1.3. KURSGRUPPEN

Das Ausbildungsangebot unterteilt sich in 4 grundsaetzliche Gruppen.

- o Anwenderseminare
- o Computergrundlagen
- o Softwareausbildung
- o Hardwareausbildung

1.3.1. SOFTWAREAUSBILDUNG

1.3.1.1 SPRACHKURSE

Alle Sprachkurse, sowohl PDP-11 Assembler, als auch fuer die hoeheren Programmier-

sprachen werden ohne spezielle Betriebssystemkenntnisse unterrichtet.

1.3.1.2. BETRIEBSSYSTEME

In unseren Betriebssystemkursen vermitteln wir dem Applikationsprogrammierer und dem Benutzer die notwendigen Kenntnisse.

1.3.1.3. SYSTEMPROGRAMMIERUNG

In diesen Kursen werden die notwendigen Spezialkenntnisse zur Systemprogrammierung (z.b. Programmierung eines Device-Drivers) vermittelt.

1.3.1.4. APPLIKATIONSBEZOGENE KURSE

Spezielle fuer Prozesssteuerungsprobleme unter RSX-11M haben wir einen Kurs entwickelt, der die notwendigen Kenntnisse anhand eines Anwendungsbeispiels vermittelt.

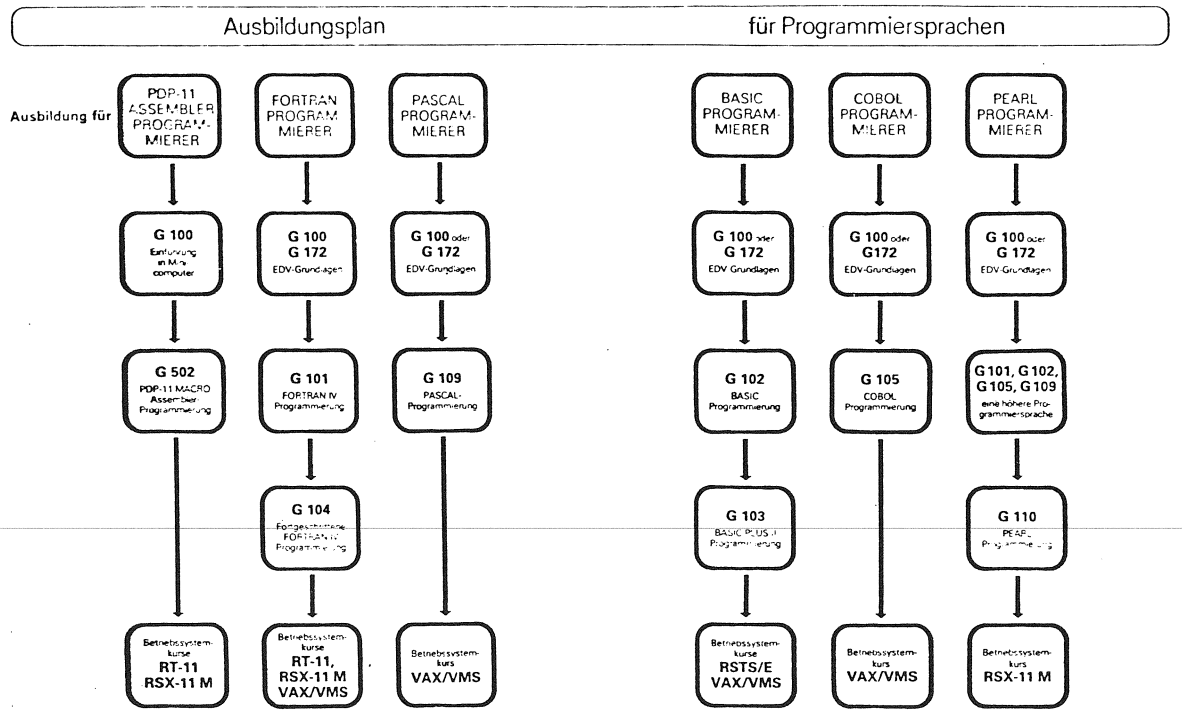
1.4. SCHULUNG-VOR-ORT

Neben den in unserem Schulungszentrum in Muenchen durchgefuehrten Kursen halten wir auf Anfrage Kurse in der Firma des Kunden ab. Dies kann bei gleichzeitiger Schulung einer Mitarbeitergruppe die wirtschaftlichere Loesung fuer die Ausbildung sein. Ein weiterer Vorteil dieser sogenannten On-Site Schulung ist, dass der Kunde die Inhalte dieser Kurse individuell auf seine Beduerfnisse abstimmen kann.

In diesen Rahmen faellt auch das sogenannte On-the-Job Training. Hier steht der Dozent zur Unterstuetzung in der Startphasen eines Projekts bei der Realisierung zur Verfuegung und vertieft vorhandenes Wissen in Bezug auf diese konkrete Anwendung.

1.5. LEHRMATERIAL

Als Alternative zur traditionellen Schulung bieten wir Lehrmaterial zum Selbststudium an. Diese Selbststudienpakete sind nach modernen paedagogischen Gesichtspunkten entwickelt und liegen in deutscher und englischer Sprache vor.



2. PEARL-11 SCHULUNG

2.1. AUSBILDUNGSKONZEPT

Das DEC-Schulungszentrum bietet seit letzem Jahr PEARL-Schulung an. Bei der Konzipierung der Ausbildung haben wir die PEARL-Kurse in unser vorhandenes Schulungsangebot und Ausbildungskonzept mit eingebunden.

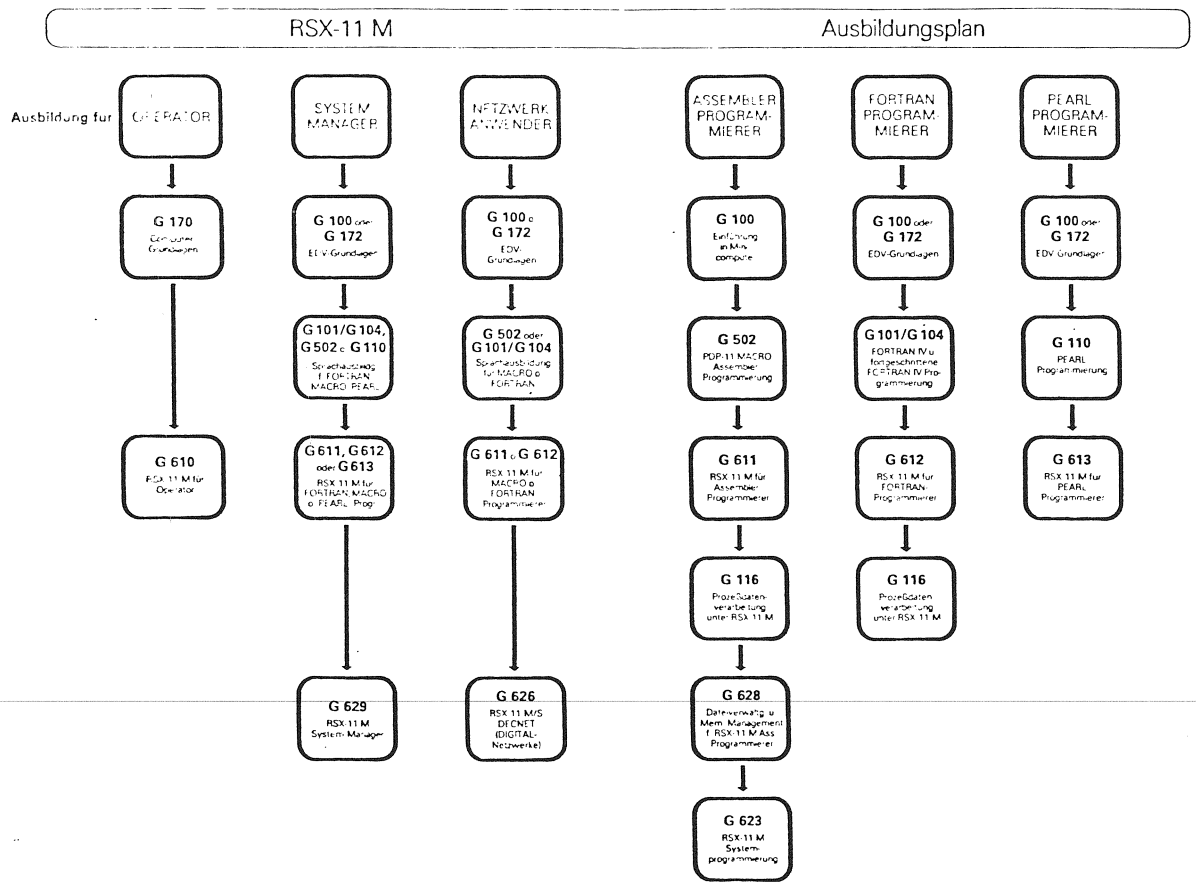
Daraus resultierend entstanden zwei Kurse, die in die bestehenden Kursgruppen integriert wurden. Die Gruppe der Sprachkurse wurde um einen reinen PEARL-Sprachkurs erweitert. Er richtet sich an Anwender-Programmierer, die keine oder wenig Erfahrung mit PEARL haben. Die Dauer dieses Kurs ist 1 Woche.

Bei den Betriebssystemkursen bieten wir einen RSX-11M Kurs fuer PEARL-Programmierer an. Auch dieser Kurs ist in den vorhandenen Ausbildungsgang integriert. Er setzt PEARL-Kenntnisse voraus und geht nur auf die implementationsspezifischen Eigenschaften und Besonderheiten unter RSX-11M ein. Dieser Kurs dauert 2 Wochen.

3. PEARL-11 SPRACHKURS

3.1. KURSBESCHREIBUNG

Obwohl dieser Kurs PEARL hinsichtlich der PDP-11 Implementierung lehrt, ist er als reiner Sprachkurs aufgebaut. Alle Themen, die spezifisch fuer eine Implementierung unter einem Betriebssystem sind, werden in einem Folgekurs behandelt. Der gelehrte Sprachumfang ist im PEARL-11 Reference Manual definiert. Unter Anleitung wird in diesem Kurs die Programmierung eines tatsaechlichen Echzeit-Projekts simuliert, um so den Stoff zu erarbeiten.



3.2. VORAUSSETZUNGEN

Bedingt durch die Dauer des Kurses (1 Woche) ist dieser Kurs sehr kompakt. Teilnehmer dieses Kurses muessen daher lange Erfahrung in einer anderen hoeheren Programmiersprache und mit Echtzeit-Anwendungen haben.

3.3. LERNZIELE

Ziel des Kurs ist es, dass der Teilnehmer

- o in PEARL-11 Programme entwickelt,
- o Datentypen und Operatoren verwendet,
- o den Programmablauf steuert,
- o I/O-Operationen programmiert

3.4. THEMEN

- o Merkmale und Eigenschaften von PEARL-11
- o Echtzeitprogrammierung
- o Tasksteuerung
- o I/O-Programmierung
- o Arithmetische Operationen
- o Reaktionen auf aeussere Ereignisse
- o Interfaces und Datenwege

3.5 KURSAUFBAU

Der Kurs folgt in Methodik und Aufbau der Ueberlegung, dass es auch beim Erlernen einer natuerlichen Sprache nicht sinnvoll ist, mit der Grammatik zu beginnen. Nach einer einfuehrenden Systembeschreibung der im Kurs als Problem angenommenen Echtzeit-Anwendung, wird der Teilnehmer daher zuerst mit einem Programm-Modul bekannt gemacht und lernt es lesen. Syntaktische Regeln werden so anhand praktischer Beispiele erlernt.

Der Kurs ist insgesamt in 7 Moduln gegliedert.

Modul 1 behandelt Definition von Datentypen und Erstellung eines Programms mit Rechenoperationen, Schleifen und Bedingungsanweisungen

Modul 2 beschaeftigt sich mit Sprachelementen zur Strukturierung von Echtzeit-Systemen

Modul 3 enthaelt Sprachelemente zur Synchronisation und Tasksteuerung

Modul 4 geht auf die I/O-Programmierung ein

Modul 5 befasst sich mit algorithmischen Operationen

Modul 6 erlaeutert die Sprachelemente, die Reaktion auf aeussere Ereignisse erlauben

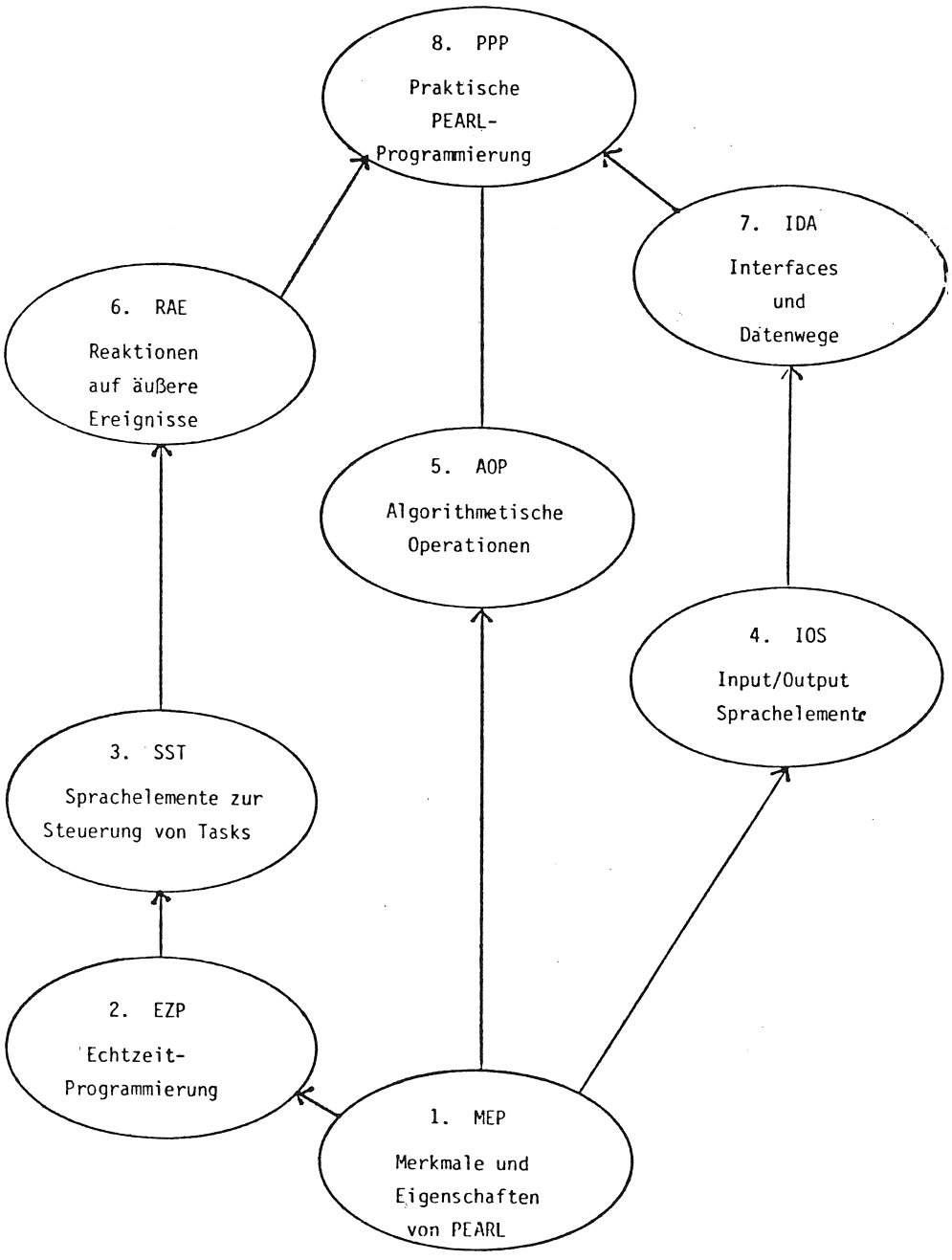
Modul 7 vermittelt Definition und Aufbau von Datenwegen zur Kommunikation mit der Peripherie

Abschliessend werden noch Moeglichkeiten zur Programmoptimierung aufgezeigt.

3.6. KURSDAUER

5 Tage

KURSAUFBAU PEARL-11 SPRACHKURS



3.7. LITERATUR

Folgende Literatur wird in dem Kurs verwendet:

- o Kursunterlage (sogenanntes Handout) in Deutsch
- o PEARL-11 Language Reference Manual

4. AUFBAU DES RSX-PEARL KURSES

4.1. KURSBESCHREIBUNG

In diesem Kurs wird die Implementierung unter RSX-11M behandelt. Der Kurs ist fuer Programmierer gedacht, die bereits PEARL-Kenntisse besitzen und Programmentwicklung unter RSX-11M durchfuehren wollen. Behandelt werden sowohl Bedienungskommandos und Dienstprogramme zur Pro-

grammentwicklung als auch Methoden des Programmtests, Optimierung der Programme unter RSX-11M und Unterstuetzung der Programme durch RSX-11M Services.

4.2. VORAUSSETZUNGEN

Teilnehmer dieses Kurses muessen bereits Erfahrung im Schreiben von Programmen in PEARL haben. Als Wissen wird der Inhalt des PEARL-11 Sprachkurses vorausgesetzt.

4.3. LERNZIELE

Ziel der Kurses ist es, dass der Teilnehmer

- o Betriebssystem-Kommandos zur Programmentwicklung benutzt,
- o PEARL-Programme unter RSX-11M entwickelt,
- o Overlay-Beschreibungen fuer PEARL-Programme lesen und erstellen kann,
- o vorhandene Teshilfen verwendet.

4.4. THEMEN

- o Bedienung und Eigenschaften von RSX-11M
- o Betriebssystem-Kommandos
- o Dienstprogramme
- o Programmentwicklung (PEARL-Compiler, TKB, LBR)
- o Testmethoden
- o Indirect Command File Processor
- o Overlay-Programmierung
- o I/O-Programmierung
- o File I/O
- o System-Direktiven
- o Benutzung der Memory Management Unit in PEARL-11
- o Eigenschaften von Interfaces in PEARL-11

4.5. KURSAUFBAU

Der Kurs ist in 10 Lehrmoduln unterteilt:

Modul 1 gibt eine allgemeine Uebersicht der Eigenschaften von RSX-11M und erlaeutert die RSX-11M Philosophie. Hier wird der Unterschied zwischen PEARL-Welt und RSX-Welt erlaeutert.

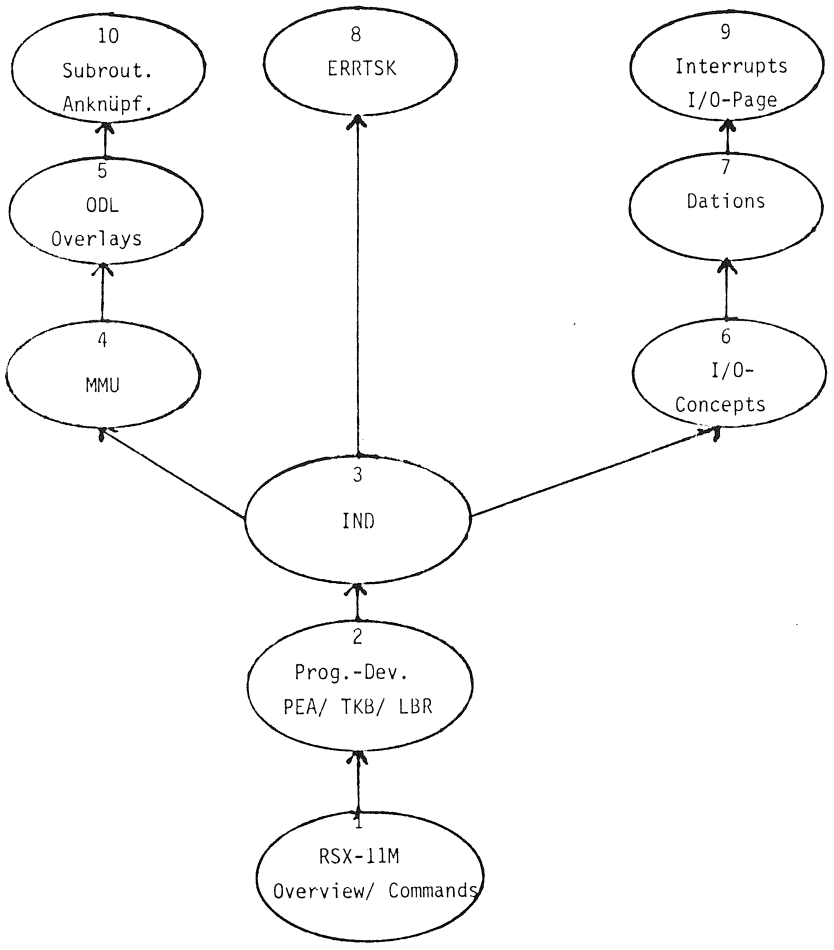
Modul 2 beschaeftigt sich mit den Schritten der Programmentwicklung, den dazu notwendigen Dienstprogrammen, wie Taskbuilder, Librarian und dem PEARL-Compiler.

Modul 3 hat den Indirect Command File Processor zum Inhalt. Es werden Moeglichkeiten zur Vereinfachung der Programmentwicklung erlaeutert.

Modul 4 geht auf die Memory Management Unit ein. Die Memory Belegung von PEARL-Programmen, Aufbau eines PEARL-Tasks und Groesse eines PEARL-Programms sowie dessen Komponenten bilden die Themen dieser Lehreinheit.

Modul 5 behandelt die Overlay-Programmierung. Besprochen wird die Overlay-Beschreibung unter RSX-11M sowie die Segmente eines PEARL-11 Tasks.

Modul 6 erlaeutert das I/O Konzept unter RSX-11M. Nach Klaerung der Grund-



lasen von In- und Output wird auf I/O unter RSX-11M und das Dation-Konzept von PEARL-11 eingesangen

Modul 7 hat System-Dations von PEARL-11 zum Thema. Es werden die unterschiedlichen Interfaces (Alphic, Basic und File-Dation) besprochen. Behandelt wird ebenfalls die Programmierung eigener Interfaces.

Modul 8 beschaeftigt sich mit der Reaktion auf Laufzeitfehler.

Modul 9 erlaeutert Interruptprogrammierung und Zugriff auf die I/O-Page. PEARL-11 sieht diese Moeglichkeit vor.

Modul 10 behandelt abschliessend die Moeglichkeit, Unterroutine und Prozeduren anzuknuepfen, die in einer anderen Sprache geschrieben wurden.

4.6. DAUER

10 Tage

4.7. LITERATUR

- o Kursunterlage (Handout) in Deutsch
- o PEARL-11 Language Ref. Manual
- o PEARL-11 Users-Guide
- o RSX-11M Task Builder Ref. Manual

o RSX-11M Utilities Ref. Manual

o RSX-11M MCR Ref. Manual

5.1. Schwerpunkte

Bei der Durchfuehrung des RSX-11M PEARL-Kurse zeigte sich, dass bei der Ausbildung Schwerpunkte gesetzt werden koennen. Bedingt durch das meistens sehr hohe Niveau der Teilnehmer (Jahrelange FORTRAN Erfahrung) war der Zeitaufwand fuer grundlegende Themen, wie RSX-Bedienung, gering. Die ungewohnte Programmentwicklung (Compiler produziert Libary-File) und aufwendigeres bilden des Tasks (TKB) verursachte schon gewissen Zeitaufwand und bedurfte der Erlaeuterung. Ein weiterer Schwerpunkt liegt auf der Klaerung des Unterschieds zwischen RSX-Welt und PEARL-Welt.

Die Tatsache des unterschiedlichen Echtzeitverhaltens von PEARL-Tasks und RSX-Tasks setzte ebenfalls einen Schwerpunkt bei diesem Kurs.

5.2. ERFAHRUNGEN BEIM SPRACHUNTERRICHT

Beim Sprachunterricht zeigte sich die absolute Notwendigkeit, dass der PEARL-Anfaenger bereits Erfahrung in einer hoeheren Programmiersprache und gutes Verstaendnis fuer Echtzeit-Anwendungen hat. Der ueberwiegende Teil der Teilnehmer hatte FORTRAN-Erfahrung.

Daraus folgten in erster Linie Schwierigkeiten bei der Umstellung auf die zu starr empfundene Syntax von PEARL. Als unhandlich wurde von den FORTRAN-Programmierern die Handhabung des I/Os in PEARL empfunden.

5.3. KURSNACHFRAGE

Die Nachfrage nach reiner Sprachschulung muss im Augenblick noch als gedaempft bezeichnet werden. Erfreulich war die Teilnehmerzahl und die Nachfrage fuer die RSX-11M PEARL-Schulung.

THOMAS HÄNSSGEN
c/o DIGITAL EQUIPMENT GMBH
- SCHULUNGSZENTRUM -
WALLENSTEINPL. 2
8000 MÜNCHEN 40

PEARL in Vorlesungen und Praktika an der FH Bielefeld

Prof. Dr. L. Frevert, Bad Salzuflen

Zusammenfassung

Für die Einführung in Basis PEARL wurde ein Skript mit 51 Beispiel-Programmen verfaßt. PEARL wurde in einer Betriebssystem-Vorlesung für die Simulation von Strategien und Koordinationsmethoden verwendet, z.B. für die Simulation von Monitoren durch PEARL Module. Im Praktikum Prozeßdatenverarbeitung entwickeln die Studenten Module eines ziemlich umfangreichen Programmes. Die Modul-Schnittstellen wurden spezifiziert mit Hilfe einer neuartigen Methode zur Top-down-Spezifikation, -Entwicklung und -Dokumentation von Programmen.

Schlüsselworte: Basis PEARL, Betriebssystem, Monitor, Entwurfs-Methode, Dokumentation

Abstract

For teaching of Basis PEARL a script containing 51 example programs was written. PEARL has been used in a lecture about operating systems for simulation of monitors by PEARL moduls. Within a practical course in process control, modules of a rather large program are to be developed by students. The module interface have been specified using a novel top down method for program specification, development and documentation.

Key words: Basic PEARL, operating systems, monitors, program design, documentation

Die Fachbereiche Elektrotechnik und Maschinenbau der FH Bielefeld betreiben gemeinsam ein Prozeßrechenzentrum, das mit einem EPR 1300-System (Hauptrechner und Satellitenrechner) der Firma Krupp-Atlas-Elektronik ausgerüstet ist. Die Verfügbarkeit von PEARL hatte bei der Auswahl des Systems eine nicht unwesentliche Rolle gespielt. Es handelt sich um Basis-PEARL, das inzwischen um einige Elemente aus full PEARL erweitert worden ist.

Das größte Hindernis für die Verwendung von PEARL in der Vorlesung und im Praktikum "Prozeßdatenverarbeitung" bildete zunächst das Nichtvorhandensein eines preisgünstigen Lehrbuches über Basis-PEARL. Ich habe deshalb als erstes ein Skript verfaßt, das im Offset-Druck vervielfältigt wurde und den Studenten zum Selbstkostenpreis von DM 5.- zur Verfügung gestellt wurde. Trotz einer Gesamtauflage von 250 Exemplaren ist es inzwischen vergriffen.

Bei der Abfassung des Skriptes wurden die einzelnen Sprachelemente von PEARL grundsätzlich anhand kompletter Beispiel-Programme erläutert - insgesamt 51. Sie sind in einem Begleitheft zusammengefaßt, sodaß beschreibender Text und Programm gleichzeitig aufgeschlagen werden können. Die Programme sind so geschrieben, daß durch möglichst geringfügige Änderungen das nächste aus dem vorhergehenden abgeleitet werden kann. Das erste Beispiel liest z.B. eine Zeichenkette von der Tastatur ein und gibt sie auf dem Bildschirm aus, das zweite liest zwei Ketten ein, kateniert sie (Ausdruck), weist das Ergebnis einer Variablen zu und gibt diese aus, das dritte vergleicht die Ketten (bedingte Anweisung), das vierte Beispiel weist das Ergebnis des Vergleichs einer Bit-Variablen zu (Einführung des Datentyps BIT), usw.. Die Studenten können deshalb die Programme mit relativ wenig Aufwand selbst ausprobieren.

Da ihnen dabei in der Regel die Prozeßperipherie des Rechners nicht zur Verfügung steht, wird ihnen Parallelarbeit, Einplanungen, Synchronisation usw. nicht mit Beispielen aus der Prozeßdatenverarbeitung erläutert, sondern durch Tasks, die Personen repräsentieren, die z.B. ein Badezimmer zusammen oder nacheinander benutzen wollen.

Das Verfahren, Systeme, die Nebenläufigkeiten enthalten, mit Hilfe von PEARL-Programmen zu simulieren, läßt sich übrigens ausgezeichnet verwenden, um die Auswirkungen verschiedener Betriebssystem-Strategien zu demonstrieren. In fast jedem Lehrbuch über Betriebssysteme wird das Problem der 5 Philosophen behandelt, denen nur 5 Gabeln zur Verfügung stehen, die aber zum Essen je zwei Gabeln benötigen. Wenn sie alle gleichzeitig eine Gabel nehmen, müssen sie verhungern

- klassisches Beispiel für eine Verklemmung. Eine mögliche Abhilfe scheint darin zu bestehen, daß ein Philosoph, der essen will, entweder zwei Gabeln gleichzeitig oder garkeine nimmt. Natürlich läßt sich mit Hilfe theoretischer Überlegungen zeigen, daß es auch bei dieser Strategie vorkommen kann, daß ein Philosoph nie gleichzeitig in den Besitz beider Gabeln kommen kann; diese Überlegungen lassen sich jedoch sehr schön konkretisieren, wenn man das Verhalten der Philosophen durch ein PEARL-Programm simuliert.

Um beim Nutzen von PEARL für Betriebssystem-Vorlesungen zu bleiben: In der Literatur und in Diskussionen wird gegenüber dem Gebrauch von Semaphoren immer wieder der Einwand gemacht, daß er Programme anfällig für Verklemmungen macht, deren Möglichkeit nur schwer erkannt werden kann, und stattdessen der Gebrauch von Monitoren (wie in CONCURRENT PASCAL) oder anderer Verfahren, z.B. von Rendezvous (Ada) empfohlen.

Ich halte diese pauschale Kritik für ebensowenig gerechtfertigt, wie ein pauschales Verbot von GOTO oder von Pointern - beide Sprachelemente entsprechen sich in etwa in ihren Auswirkungen auf die Integrität von Programmen. Diese Kritiker übersehen nämlich, daß das Modul-Konzept von PEARL dazu verwendet werden kann, die Zuverlässigkeit von Programmen zu erhöhen. Man braucht sich nur an die Regel zu halten, die Koordination oder Synchronisation von Tasks in eigens geschriebene Moduln zu verlegen. Ich möchte dies an einem Beispiel verdeutlichen: beim Leser-Schreiber-Problem dürfen beliebig viele Tasks gleichzeitig aus einem Datenbereich lesen, während immer nur eine Task die Daten schreibend verändern darf, und zwar nur dann, wenn nicht gleichzeitig gelesen wird. In full PEARL könnte diese Aufgabe mit Hilfe einer Bolt-Variablen gelöst werden, die allen Tasks bekannt ist. Es läge dann in der Verantwortung der Programmierer, vor und nach jedem Lesen eine ENTER- bzw. LEAVE-Anweisung zu programmieren, und vor und nach jedem Schreiben eine RESERVE- bzw. FREE-Anweisung. Offensichtlich ist bei diesem Ansatz schwer sicherzustellen, daß diese Programmiervorschriften immer befolgt werden. Besser ist es, sowohl die Daten als auch die Bolt-Variable in einem eigenen Modul zu "verstecken" und nur über globale, reentrante Prozeduren LESEN und SCHREIBEN auf die Daten zuzugreifen. Am Anfang und Ende der Prozedur LESEN müßte ENTER DATENBOLT, bzw. LEAVE DATENBOLT stehen, und am Anfang und Ende der Prozedur SCHREIBEN RESERVE DATENBOLT, bzw. FREE DATENBOLT. Da die Programmierer der zugreifenden Tasks jetzt nur noch über die Aufrufe CALL LESEN (.....) und CALL SCHREIBEN (.....) an die Daten herankommen, ist eine völlig sichere Programmierung gewährleistet.

Bei Verwendung von Basis-PEARL läßt sich das Problem mit Hilfe zweier Semaphoren KOORDINATOR und EXKLUSIVZUGRIFF und einer Variablen ZAHLDERLESER lösen, die mit 1,1 bzw. 0 initialisiert werden müssen.

Statt ENTER muß stehen

```
            REQUEST KOORDINATOR;
              ZAHLDERLESER:= ZAHLDERLESER + 1;
              IF ZAHLDERLESER == 1 THEN
                 REQUEST EXKLUSIVZUGRIFF;
              FIN;
            RELEASE KOORDINATOR;
statt LEAVE:
            REQUEST KOORDINATOR;
              ZAHLDERLESER:= ZAHLDERLESER - 1;
              IF ZAHLDERLESER == 0 THEN
                 RELEASE EXKLUSIVZUGRIFF;
              FIN;
            RELEASE KOORDINATOR;
statt RESERVE:
            REQUEST EXKLUSIVZUGRIFF;
statt FREE:
            RELEASE EXKLUSIVZUGRIFF;
```

Das Beispiel zeigt, daß es möglich ist,
1) PEARL-Mudule zu schreiben, die den MONITORen von Concurrent PASCAL entsprechend - der Witz dabei ist, daß das Leser-Schreiber-Problem in Concurrent PASCAL nicht lösbar ist, weil sich grundsätzlich nur eine Task in einem PASCAL-Monitor aufhalten darf;
2) andere Koordinations-Methoden mit Semaphoren und Merkvariablen zu simulieren - es ist eine hübsche Übungsaufgabe, dafür zu sorgen, daß kein neuer Leser zugreifen darf, solange ein Schreiber schreiben möchte.

Eine andere hübsche Übungsaufgabe besteht darin, beliebig viele Tasks über Botschaften zu koordinieren, unter Verwendung von 3 Semaphoren.

Ich muß zugeben, daß ich bisher noch nicht sehr systematisch untersucht habe, inwieweit sich in der Betriebssystem-Literatur angegebene Koordinations-Verfahren mit PEARL simulieren und untersuchen lassen - mir scheint das jedoch ein lohnendes Feld zu sein.

Eines der Probleme beim Unterricht in Datenverarbeitung ist, die Studenten vom Nutzen der Methoden

zu überzeugen, die bei der Erstellung großer Programme unumgänglich sind. Die herkömmlichen Übungsaufgaben müssen notwendigerweise im Umfang beschränkt sein und lassen sich notfalls auch durch unsystematisches Vorgehen lösen. Ich bin deshalb dazu übergegangen, in den Praktika Aufgaben zu stellen, die Teilaspekte eines größeren Problemes lösen und aufeinander aufbauen, sodaß am Ende des Praktikums ein relativ umfangreiches Programm entstanden ist. Das erfordert eine sehr sorgfältige Aufgliederung des Gesamtproblems in Einzelteile, die vom Dozenten vorgenommen werden muß, weil den Studenten die dazu nötige Erfahrung fehlt. Wir besitzen in unserem Praktikum eine Modelleisenbahn, die vom Rechner gesteuert werden kann. Dabei wird der jeweilige Ort der Lokomotiven aufgrund von Interrupts ermittelt, die durch Gabellichtschranken erzeugt werden.

Die Gesamtaufgabe besteht darin, einen sicheren Betrieb der Anlage zu gewährleisten, den jeweiligen Anlagenzustand und die Zugorte auf einem alphanumerischen Sichtgerät darzustellen, Ereignisse, wie das Stellen von Weichen und Signalen zu protokollieren, usw.. Die Fahrwege der Züge und Aufenthaltsdauern vor Signalen sollen ohne Rücksicht auf das Vorhandensein anderer Züge festgelegt werden können. Darüber hinaus sollen alle Programme weitgehend unabhängig von der Topologie der speziellen Anlage geschrieben werden; die Topologie wird in einer Datenbank beschrieben, auf die die einzelnen Programmteile über Zugriffsprozeduren zugreifen.

Die einzelnen Teilaufgaben beginnen stets: "Für die rechnergesteuerte Modelleisenbahn ist ein PEARL-Modul zu entwickeln und zu testen, der folgende Prozeduren enthält: ....."

Das gesamte Programm besteht aus 10 Moduln, von denen die Studenten-Gruppen etwa 5 entwickeln; die übrigen sind von mir programmiert worden und werden im begleitenden Seminar besprochen, wie die Prinzipien, nach denen die Aufteilung vorgenommen wurde. Vorsichtshalber habe ich auch die anderen Moduln in Reserve, um auch den Studentengruppen ein abschließendes Erfolgserlebnis beim Integrationstest vermitteln zu können, bei denen sich der eine oder andere Modul als noch fehlerhaft erweist.
Um die Schnittstellen zwischen den Moduln festlegen zu können, wurden die wichtigsten Moduln mit einem Top-down-Verfahren in Pseudocode programmiert. Das Verfahren ist so angelegt, daß bei weitergehender Verfeinerung die Ebene von PEARL-Anweisungen erreicht wird; ein eigens geschriebener Preprozessor wandelt die so erhaltenen Programm-Entwürfe in kompilierbare Programme um. Ich möchte das Verfahren an einem Beispiel erläutern:

Einer der Moduln enthält die Tasks und Unterprogramme, die dafür sorgen, daß einerseits die Weichen und Signale für den Fahrweg eines Zuges richtig gestellt werden, andererseits die Züge aber auch nicht zusammenstoßen. Der Modul hat folgenden Aufbau:

```
/* ZENTRALE SICHERHEITSSTEUERUNG
*/ MODULE BAHNU; /*
    #1    SCHNITTSTELLEN ZU ANDEREN MODULN
          (SPEZIFIKATIONEN)
    #2    VARIABLE
          (DEKLARATIONEN)
    #3    PROZEDUREN
    #4    TASKS
*/ MODEND; /*
```

Da das Ziel der Programmierung darin besteht, die Schnittstellen zu anderen Moduln zu gewinnen, können diese noch nicht spezifiziert werden; desgleichen werden sich die benötigten Variablen und Unterprogramme erst aus dem Entwurf der Tasks ergeben. Die weitere Entwicklung beginnt deshalb mit der Verfeinerung von #4, für die die obige Zeile die Überschrift darstellt. Um Überschriften und zugehörige Verfeinerungen leicht unterscheiden zu können, werden bei ersteren die zugehörigen Kennziffern eingerückt, bei letzteren ganz an den Rand geschrieben. Die Verfeinerung von #4 lautet:

```
#4
    #41  FUER JEDE GABEL-LICHTSCHRANKE EXISTIERT
         EINE TASK, DIE IN EINER PROZEDUR BAHNUI
         EINGEPLANT WIRD, SODASS SIE DURCH DEN
         GABEL-LICHTSCHRANKEN-INTERRUPT AKTIVIERT
         WIRD.
    #42  FUER JEDEN FAHRWEG EXISTIEREN ZWEI TASKS;
         DIE EINE SORGT DAFUER, DASS DIE WEGELE-
         MENTE FUER DIE FORTSETZUNG DES FAHRWEGES
         EXKLUSIV RESERVIERT, DIE WEICHEN UND SIG-
         NALE RICHTIG GESTELLT WERDEN. DIE ANDERE
         GIBT BEREITS DURCHFAHRENE WEGELEMENTE FREI.
         BEIDE TASKS WERDEN BEI BEDARF DURCH DIE
         GABEL-LICHTSCHRANKENTASKS AKTIVIERT.
```

Hier soll uns zunächst die Verfeinerung von #4 interessieren:

```
#41
    #411 TASKS
    #412 ALLEN TASKS GEMEINSAME BEARBEITUNGSPROZE-
         DUR
```

Die Verfeinerung von ╫ 411 besteht aus PEARL-Code:

```
╫411     */ SENSORTASK1: TASK;
              CALL MELDUNGPRUEFEN(1);
            END;
            :
            SENSORTASK11: TASK;
              CALL MELDUNGPRUEFEN(11);
            END /*
```

(unsere Anlage hat insgesamt 11 Gabel-Lichtschranken)

Die Verfeinerung ╫412 muß weiter verfeinert werden:

```
╫412
        */ MELDUNGPRUEFEN: PROC(SENSORNR FIXED)
                                        REENT; /*
  ╫4121 STELLE FEST, OB AM BETREFFENDEN SENSOR
        DAS DURCHFAHREN EINES ZUGES ERWARTET
        WURDE
  ╫4122 IF MELDUNG ERWARTET
  ╫4123 THEN HANDLE ENTSPRECHEND
  ╫4124 ELSE MELDE FEHLER; FIN
        */ END; /*
```

Die Verfeinerung von ╫4121 besteht aus dem Aufruf einer Zugriffsprozedur zur zentralen Datenbank, in der Topologie und Zustand der Anlage notiert sind; aus der Pseudocode-Programmierung hier ergibt sich, daß in dieser Datenbank notiert werden muß, ob ein Zug an einer Gabel-Lichtschranke erwartet wird.

```
╫4121    */ FAHRWEGNR:=HOLFAHRWEGNR(SENSORNR); /*
╫4122    */ IF FAHRWEGNR/= 0/*
```

FAHRWEGNR ist eine lokale Variable in MEDLUNG-PRUEFEN. Der zugehörige Pseudocode läßt sich leicht an entsprechender Stelle einfügen.

```
  ╫4120 LOKALE VARIABLE
```

Der Programmentwurf läßt sich fortsetzen mit

```
╫4123    */ THEN /*
  ╫41231 VERFOLGE FAHRWEG BIS ZUM NAECHSTEN
         SENSOR ODER SIGNAL UND TRAGE DORT DIE
         FAHRWEGNR EIN
  ╫41232 IF SIGNAL
  ╫41233 THEN AKTIVIERE FORTSETZUNGSTASK; FIN
  ╫41234 AKTIVIERE FREIGABETASK
```

usw..

Wie man sieht, braucht in einem so gewonnenen vollständigen Programmentwurf nur jeweils die zugehörige Verfeinerung hinter einem Pseudocode eingeordnet zu werden, um ein PEARL-Programm zu erhalten, in dem die Pseudocodestücke als Kommentare stehen. Genau diese Umordnung der Zeilen-Reihenfolge bewerkstelligt der oben erwähnte Preprozessor, der außerdem den Entwurf auf Vollständigkeit und richtigen Gebrauch der Kommentar-Symbole untersucht. Ich habe schon erwähnt, daß die Studenten im Praktikum die Moduln des Modellbahn-Programmes einzeln entwickeln und testen, bevor sie zum Gesamtprogramm integriert werden. Dazu muß jeder Modul mit Programmteilen versehen werden, die nur zum Modultest benötigt werden - z.B. zur Simulation der Schnittstellen zu anderen Moduln. Diese Programmteile lassen sich bei Programmierung mit dem eben beschriebenen Verfahren praktisch vollautomatisch aus den fertigen Moduln entfernen: der Preprozessor übergeht auf Wunsch alle Verfeinerungen, die mit ╫ % beginnen, und ordnet sie nicht in das kompilierbare Programm ein. Diese Eigenschaft des Preprozessors läßt sich übrigens auch dafür ausnützen, top-down gegliederte Spezifikationen mit in das Entwurfsdokument zu integrieren. Auf diese Weise sind Spezifikationen, Programmentwurf und Testschnittstellen in einem einzigen Dokument enthalten. Umgekehrt werden Programmteile, die für den Test überflüssig, für das Gesamtprogramm jedoch notwendig sind (z.B. Spezifikationen von Schnittstellen zu anderen Moduln) mit ╫§ gekennzeichnet.

Seit einiger Zeit habe ich das Verfahren auch für die Studenten freigegeben. Die Erfolge sind ermutigend; die Akzeptanz ist groß, weil der zusätzliche Lern- und Schreibaufwand gering ist und die Vorteile - schnellere Programmentwicklung und übersichtlichere Dokumentation - sofort ersichtlich sind.

Anschrift des Autors

Prof.Dr.L.Frevert
Ostersiek 29, 4902 Bad Salzuflen
Telefon: 05222/10126

PEARL-Ausbildung im Studiengang Elektrotechnik an der Universität (GH) Paderborn

Dr.-Ing. Bernd Reißenweber, Paderborn

Zusammenfassung:

Die Ausbildung besteht aus einer zweisemestrigen Vorlesung und Übungen mit einem BBC-PEARL-System. Vorlesungsinhalt und die Art, wie in der Vorlesung vorgegangen wird, werden beschrieben. Von den Übungsaufgaben wird die Führung eines kleinen Prozesses ausführlicher erläutert, und es wird auf die Erfahrungen mit dem BBC-PEARL-System bei der Durchführung der Übungen eingegangen. Den Abschluß bilden Überlegungen, wie man die PEARL-Ausbildung intensiver und nachhaltiger gestalten könnte.

1. Einführung

Für die PEARL-Ausbildung stehen innerhalb der Universität Paderborn zwei Systeme zur Verfügung:

a) Siemens 330 mit Basis-PEARL im Fachbereich Mathematik-Informatik

b) BBC DP 1000 mit BBC-PEARL bzw. PAS2 im Fachgebiet Prozeßautomatisierung des Fachbereichs Elektrotechnik.

Auf dem Siemens-System werden die Studenten des sechssemestrigen Studienganges, auf dem BBC-System die Studenten des achtsemestrigen Studienganges ausgebildet. Hier soll über die Ausbildung am BBC-Rechner berichtet werden.

2. Art der Veranstaltung

Die Studenten sollen in die Lage versetzt werden, ein vollständiges PEARL-Quellprogramm schreiben und am Prozeßrechner zur fehlerfreien Ausführung bringen zu können. Dazu werden in einer normalen Vorlesung die einzelnen Sprachelemente von PEARL und die Regeln zum Schreiben eines PEARL-Programmes behandelt. Zusätzlich werden Übungsaufgaben ausgegeben. Die Studenten müssen das zur Lösung der Aufgabe notwendige Programm entwickeln und in Übungsstunden am Rechner austesten.

Die gesamte Veranstaltung wird als Wahlpflichtfach durchgeführt, d.h. die Studenten nehmen freiwillig daran teil. Bisher waren es pro Semester etwa 15 - 25 Hörer aus dem 5. oder höheren Semestern. Als Pflicht-Programmiersprache haben diese Studenten bereits Assembler bzw. seit letztem Jahr FORTRAN gelernt. Diese Vorbildung wirkt sich stark auf das Verhalten der Studenten in Vorlesung und Übung aus, so daß man insbesondere in der Ausgestaltung der Vorlesung darauf Rücksicht nehmen muß.

3. Aufbau der Vorlesung und Übung

Im Fachgebiet Prozeßautomatisierung beschäftigen wir uns bevorzugt mit dem Entwurf und der Realisierung von Regelungs- und Steuerungsalgorithmen und mit der Simulation dynamischer Systeme. Dazu muß man - was die Algorithmen anbelangt - numerisch Integrieren, Lösungen von Differentialgleichungen bestimmen, Nullstellen suchen und Matrizen manipulieren. Hinzu kommen Echtzeitaufgaben wie paralleler Lauf von verschiedenen Regelalgorithmen, gleichzeitiger Dialog über Bedienschreibmaschine, gleichzeitiges Erzeugen von Protokollen aus dem Drucker. Auf diese Aufgaben konzentrieren sich dann auch die PEARL-Vorlesung und die Übung.

Die Vorlesung läuft über zwei Semester. In PEARL I (3 Wochenstunden im Wintersemester)

wird der algorithmische Teil von PEARL einschließlich Ein- und Ausgabe über Standardperipherie gebracht.

Die Vorlesung beginnt mit einer kurzen Beschreibung des Prozeßrechners und seiner Peripherie und einer Beschreibung der Systemprogramme, die bei der Programmentwicklung und Programmausführung notwendig sind.

Nächster Punkt ist die Formulierung einer Aufgabe aus der numerischen Mathematik. Der Integralsinus

$$Si(x) = \int_0^x \frac{\sin t}{t}\, d$$

soll für eine Reihe von x-Werten durch verschiedene Integrationsverfahren und durch Reihenentwicklung numerisch berechnet werden. Durch einige Zusatzforderungen ist dieses Beispiel so angelegt, daß man die wesentlichen Sprachelemente, die PEARL im algorithmischen Teil bietet, damit demonstrieren kann.

Die Lösung dieser Aufgabe wird zunächst in Form von Programmablaufplänen dargestellt. Gleichzeitig werden Kopien des vollständigen PEARL-Quellprogrammes an alle Hörer verteilt.

Dann werden in der Vorlesung nacheinander die Datenelemente, Ausdrücke, Anweisungen, Ein- und Ausgabe über Standardperipherie, Prozeduren und ON-Blöcke behandelt. Dabei wird immer in drei Schritten vorgegangen:

a) An Hand des Integralsinus-Beispiels wird gezeigt, warum das gerade betrachtete Sprachelement überhaupt notwendig ist und wozu man es benutzen kann.

b) Die allgemeinen Regeln für dieses Sprachelement werden beschrieben. Dabei werden gut überschaubare Formulierungen angestrebt. Auf die Metasprache wird weitgehend verzichtet, und es werden auch nicht alle Möglichkeiten, die nach Sprachbeschreibung [1] vorhanden sind, behandelt.

c) Für jede Regel werden mehrere Beispiele hinzugefügt.

Parallel zu den letzten Vorlesungsstunden werden drei Übungsaufgaben gestellt, nämlich Nullstellensuche mit Newton-Verfahren und Regula falsi, Lösung einer Differentialgleichung 1. Ordnung mit Runge-Kutta-Verfahren, Matrizenoperationen (Dialogprogramm zum Ein- und Ausgeben von Matrizen, Proceduren für Matrizenmultiplikation und Matrizeninversion).

In PEARL II (2 Wochenstunden im Sommersemester) werden die Echtzeitelemente von PEARL behandelt.

Die Vorlesung beginnt wieder damit, daß eine Aufgabe formuliert und das dazugehörige PEARL-Quellprogramm verteilt wird. Da bei den Studenten im 6. Semester nur wenig Kenntnisse zum Thema Prozeßautomatisierung vorausgesetzt werden können, wird eine einfache Meßwerterfassungsaufgabe konstruiert.

Von einem schnelleren und einem langsameren Teilprozeß sollen in verschiedenen Zeitrastern analoge Meßwerte aufgenommen und je ein Wirkungsgrad daraus berechnet werden. Wenn eine bestimmte Anzahl von Meßwerten vorliegt, soll die relative Häufigkeit des Auftretens eines bestimmten Wertes von den beiden Wirkungsgraden ermittelt und als Druckerbild (B i l d 1) ausgegeben werden. Außerdem sollen in einem festen Zeitraster die aktuellen Meßwerte auf dem Drucker ausgegeben werden.

Auch diese Aufgabe ist so angelegt, daß möglichst viele Sprachelemente, die in PEARL II behandelt werden sollen, darin vorkommen. Es werden deshalb in der gleichen Art wie in PEARL I die Themenkomplexe Ein- und Ausgabe über Prozeßperipherie und Multitasking behandelt.

So kann man an dem Meßwerterfassungsbeispiel zeigen, daß eine Echtzeitaufgabe aus mehreren asynchron nebeneinander ablaufenden Teilaufgaben besteht und daß es sinnvoll ist, jede Teilaufgabe durch eine Task mit geeigneter Priorität zu realisieren. Es ergibt sich die Notwendigkeit, Tasks starten, beenden und unterbrechen zu können. Auch die Wichtigkeit von Semas wird an diesem Beispiel deutlich, z.B. wird der Schnelldrucker von zwei Protokollierungstasks benutzt, und es soll natürlich die Protokollzeile mit den aktuellen Meßwerten nicht mitten im Druckerbild stehen.

In der PEARL II-Übung wird eine Prozeßführungsaufgabe, die im nächsten Abschnitt beschrieben wird, gestellt.

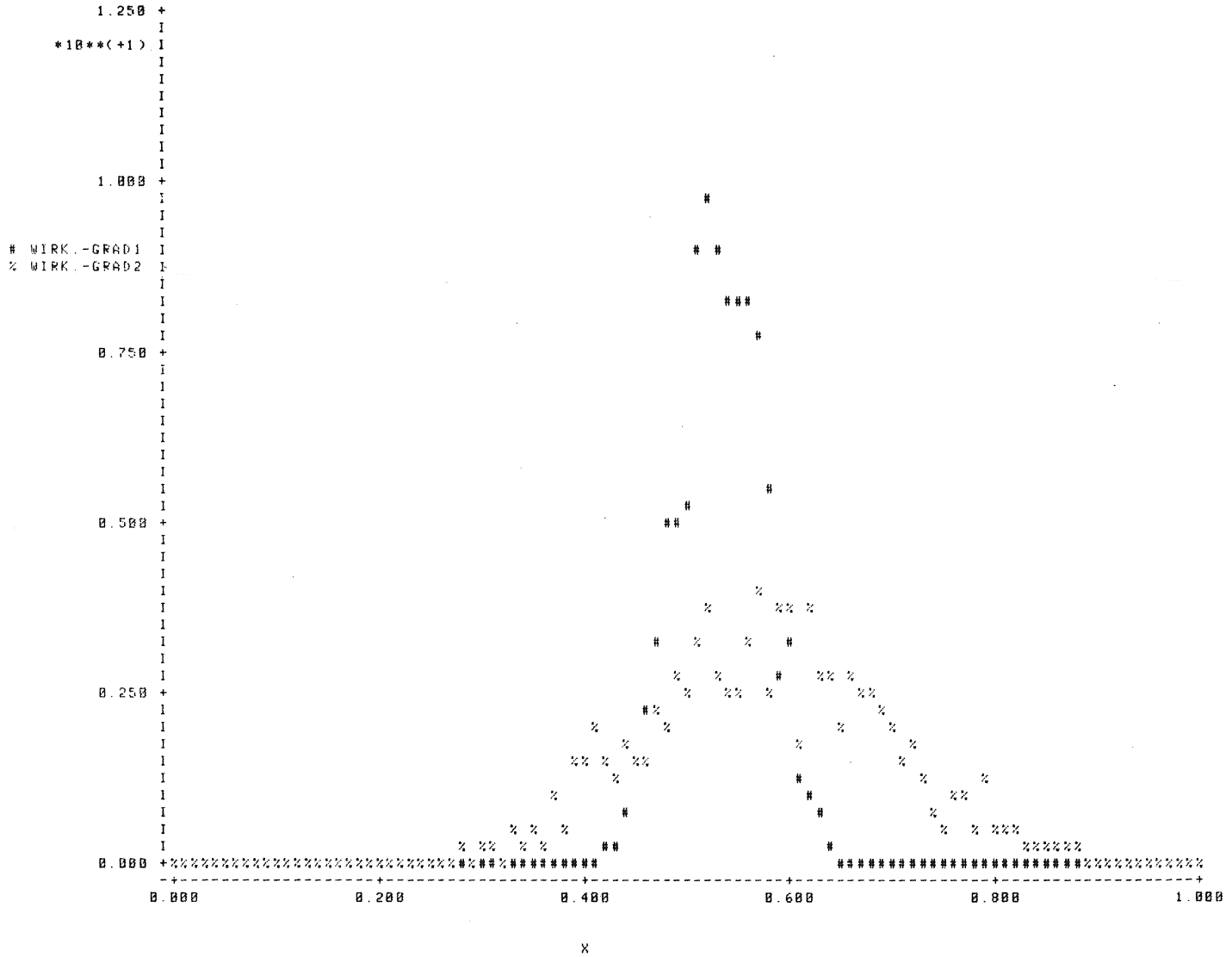


Bild 1. Relative Häufigkeit des Auftretens von bestimmten Werten des Wirkungsgrades 1 und 2 als "Druckerbild" auf einem Schnelldrucker ausgegeben

4. Erfahrungen mit dem BBC-PEARL-System bei den Übungen

In der Vorlesung wird für jede Übungsaufgabe ein Aufgabenblatt, das neben der Aufgabenstellung auch theoretische Grundlagen und Lösungshinweise enthält, verteilt. Die Studenten müssen das vollständige PEARL-Programm entwickeln und in Lochkarten ablochen. An festgesetzten Terminen haben die Studenten einen ganzen Nachmittag die Möglichkeit, ihr Programm zu übersetzen und es zum fehlerfreien Laufen zu bringen.

Dies beginnt zunächst einmal für jedes Programm mit einem, normalerweise jedoch mehreren Compilerläufen, bis der Compiler keine Fehler mehr meldet.

Der Compiler arbeitet absolut zuverlässig, er findet jeden Fehler und macht detaillierte Angaben, die die Fehlerursache sehr stark einkreisen. Zwangsläufig kann ein Fehler aber auch Folgefehler erzeugen. So entstehen in den Übungen gar nicht selten 1, 2 oder 3 Seiten lange Fehlermeldungen. Da hat man natürlich keine Lust, jeder Fehlermeldung nachzuforschen, man wird die gröbsten Fehler beseitigen und durch einen neuen Compilerlauf die Anzahl der Fehlermeldungen zu reduzieren versuchen.

Auch beim Suchen nach logischen Fehlern, die kein Compiler finden kann, wird natürlich oft so verfahren, daß man, sobald der erste Fehler gefunden ist, erst einmal neu compiliert und schaut, ob das Programm jetzt läuft.

Für den Übungsbetrieb ist es also bei der großen Anzahl von Compilerläufen sehr wichtig, daß der Compiler schnell arbeitet und daß er möglichst wenig Papier verbraucht. Damit sieht

es bei dem BBC-System nicht schön aus. Für die einfache Übungsaufgabe "Nullstellensuche" braucht er etwa 5 Minuten. Außerdem erzeugt er neben den notwendigen Ausdrucken zusätzliche 8 Seiten Papier, die für Studenten, aber auch für Mitarbeiter des Fachgebiets vielleich 10 Zeilen Information enthalten.

Als Beispiel für eine Übungsaufgabe soll die Prozeßführungsaufgabe aus PEARL II etwas ausführlicher beschrieben werden. Ein kleiner Prozeß, bestehend aus 3 Teilprozessen, wird auf einem Analogrechner realisiert und soll durch den Prozeßrechner geführt werden (B i l d 2). Der eine Teilprozeß soll durch einen PI-Regler, der andere durch ein Zweipunktglied mit Hysterese geregelt werden. Beide Regler erhalten als Eingangsgrößen die analogen Regelgrößen x_1 und x_2 und die analogen Sollgrößen w_1 und w_2. Die Sollgrößen werden als Rechteckschwingung vorgegeben und von einem Funktionsgenerator erzeugt. Aufgabe der Regler ist es, dafür zu sorgen, daß die Regelgröße x_1 möglichst gut mit der Sollgröße w_1 übereinstimmt und daß die Regelgröße x_2 möglichst gut mit der Sollgröße w_2 übereinstimmt (B i l d 3). Dazu gibt der PI-Regler die analoge Stellgröße y_1 auf den Teilprozeß 1 und der Zweipunktregler die digitale Stellgröße y_2 auf den Teilprozeß 2.

Alle Signale, die zwischen Rechner und Prozeß ausgetauscht werden, können auf einem Speicheroszillograph sichtbar gemacht und mit einem Vierkanalschreiber aufgezeichnet werden (B i l d 3).

Besonders der PI-Algorithmus kann die ihm gestellte Aufgabe gut oder schlecht erfüllen, je nachdem, wie die Parameter im Algorithmus eingestellt sind. Deshalb sollen diese Parameter während des Betriebs im Dialog über Bedienschreibmaschine geändert werden können. Dazu braucht man, da jeder Regelalgorithmus durch eine Task realisiert wird, die dritte Task. Hinzu kommt eine vierte Task, die eine Meldung auf der Bedienschreibmaschine ausdruckt, wenn der Ausgang von Teilprozeß 3 einen Grenzwert überschreitet und ein Grenzwertmelder einen Interrupt auslöst.

Da das Aufgabenblatt einige Lösungshinweise enthält und da die einzelnen Tasks relativ kurze und nahezu geradlinig durchlaufene Programmstückchen sind, treten beim Austesten dieser Übungsaufgabe nur relativ wenige logische Fehler auf. Die meisten

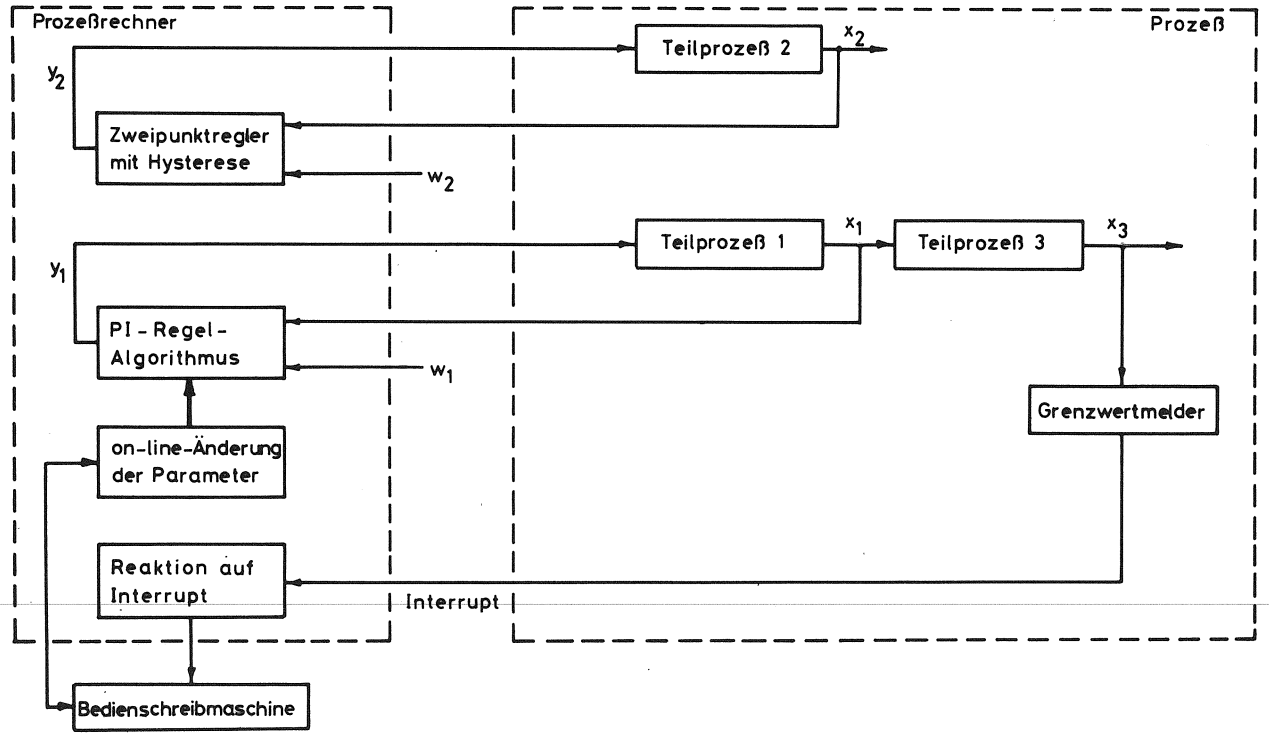


Bild 2. Führung eines einfachen Prozesses

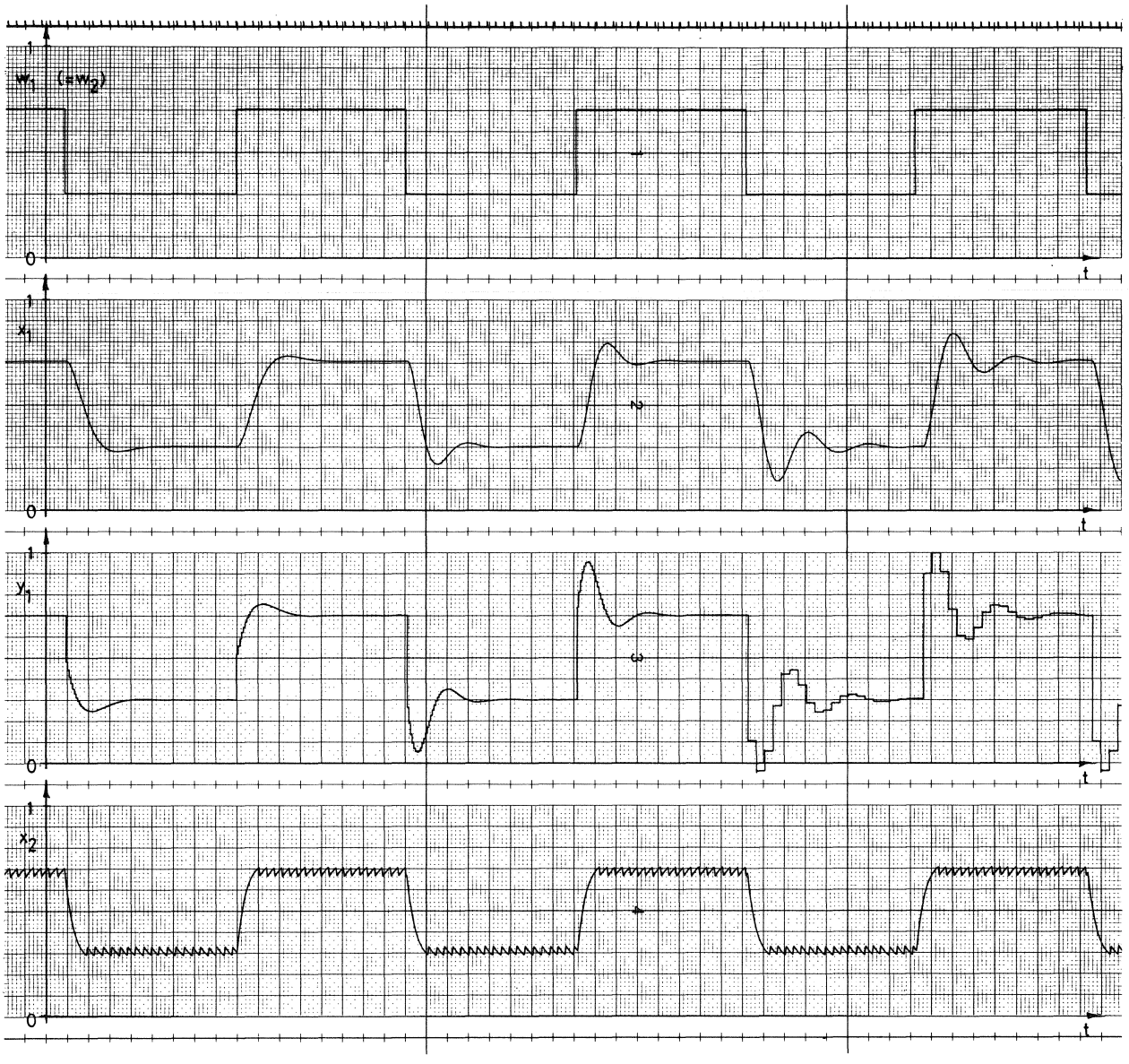


Bild 3. Zeitlicher Verlauf der interessierenden Größen bei on-line-Änderung der Parameter des PI-Algorithmus

Fehler sind von der Art, wie sie bereits in den Übungen zu PEARL I aufgetreten sind, betreffen also z.B. die Ein- und Ausgabe über Standardperipherie. Mit dem Multitasking gibt es so gut wie gar keine Schwierigkeiten. Fehler treten eher bei der Ein- und Ausgabe über Prozeßperipherie auf, da hier PAS2 recht deutlich von der Norm abweicht und einige Umständlichkeiten enthält.

Insgesamt betrachtet kann man sich auf das BBC-PEARL-System bei den Übungen vollständig verlassen. Systembedingte Fehler treten nicht auf. Letztlich kam noch jedes Programm eines Studenten zum fehlerfreien Laufen. In hartnäckigen Fällen hilft da die Bedientask, die ein gutes Testhilfsmittel ist.

5. Fortsetzung der Ausbildung

Von den Studenten wird immer wieder beklagt, daß sie die in der PEARL-Vorlesung und-Übung erworbenen Kenntnisse innerhalb der Hochschule nicht nutzen können, weil kein PEARL-System frei zugänglich ist. Außerdem zeigen die Übungen in PEARL II, daß die Studenten die in PEARL I erlernten Fähigkeiten wohl schon vergessen haben, wenn sie ein halbes Jahr später die PEARL II-Übung absolvieren.

Um die Ausbildung also intensiver und nachhaltiger werden zu lassen, muß den Studenten die Möglichkeit gegeben werden, nach Wunsch auf ein PEARL-System zugreifen zu können.

Unser Prozeßrechner eignet sich dazu nicht, da die Bedienung der zahlreichen verwendeten Systemprogramme so kompliziert ist, daß man nicht jeden Studenten darin einführen kann. Deshalb werden bei uns zur Zeit nur die Studenten, die den Prozeßrechner im Rahmen ihrer Diplom- oder Studienarbeit benutzen, in die Lage versetzt, das System selbständig zu bedienen.

Für die Ausbildung wünschenswert ist deshalb ein PEARL-System, das auf einem Mikroprozessor-Entwicklungssystem mit Floppy-Disk eigenständig lauffähig ist. Der Compiler soll ebenfalls auf dem Mikrorechner laufen, er soll möglichst schnell sein und minimalen Papierverbrauch haben. Das gesamte System soll möglichst einfach zu handhaben sein, so daß ein Student an Hand einer Bedienungsanleitung und nach einer Einführung innerhalb der Übungen das System selbständig bedienen kann.

Schrifttum

[1] BBC-PEARL-Subset, Sprachbeschreibung, Ausgabe 6.0, Juli 1977

Anschrift des Autors:

Reißenweber, Bernd
Universität (GH) Paderborn
Fachbereich Elektrotechnik
Fachgebiet Prozeßautomatisierung

Pohlweg 47 - 49

4790 Paderborn

Telefon: (05251) 60 30 05

Erfahrungen, Konzept und Ablauf verschiedener PEARL-Schulungen am IVD Stuttgart

Thomas Roehrich

Abstract. The purpose of this article is to describe the several PEARL teaching activities of the Institut fuer Verfahrenstechnik und Dampfkesselwesen. In addition an account is given of the performance and acception of the first course "Introduction to the programming language PEARL" which was organized in cooperation with the VDI- Bildungswerk and the PEARL Association.

Keywords. Process Control, teaching PEARL, VDI- Bildungswerk, PEARL course

Zusammenfassung. Im vorliegenden Artikel werden die am Institut fuer Verfahrenstechnik und Dampfkesselwesen regelmaessig durchgefuehrten PEARL- Lehrveranstaltungen vorgestellt. Zusaetzlich wird ueber die Durchfuehrung und Resonanz des ersten, in Zusammenarbeit mit dem VDI Bildungswerk und dem PEARL-Verein, abgehaltenen Kurses "Einfuehrung in die Programmiersprache PEARL" berichtet.

Schluesselworte. Prozessdatenverarbeitung, PEARL- Kurs, PEARL- Lehrveranstaltung, Erfahrungen, VDI- Bildungswerk.

1. EINFUEHRUNG

Die Abteilung Stromerzeugung und Automatisierungstechnik des Institutes fuer Verfahrenstechnik und Dampfkesselwesen (kurz IVD) ist seit 1973 durch verschiedene Arbeiten an der PEARL Entwicklung beteiligt.

In der Lehre bieten wir seit 1976 im Rahmen des Hauptfaches "Prozessdatenverarbeitung" fuer die Studienrichtung Maschinenwesen an der Universitaet Stuttgart u.a. jaehrlich im SS die zweistuendige Vorlesung "Prozessrechnersprachen und -programmierung" an. Ein Schwerpunkt dieser Vorlesung war von Anfang an die Vermittlung der Prozessrechnersprache PEARL. Zusaetzlich bieten wir im Rahmen des Hauptfachpraktikums zwei PEARL Praktikumsversuche an.

Seit Sommer 1981 beteiligen wir uns als ausfuehrende Stelle an den vom PEARL Verein in Zusammenarbeit mit dem VDI- Bildungswerk angebotenen PEARL Kursen. So wurde vom 14.-18. Sept. 1981 der erste dieser Kurse an unserem Institut durchgefuehrt.

Im folgenden wird ueber diese PEARL Lehraktivitaeten berichtet.

2. DIE VORLESUNG "PROZESSRECHNERSPRACHEN UND PROGRAMMIERUNG"

Bis einschliesslich Sommersemester 1980 war die Vorlesung wie folgt aufgebaut:

a) Grundlagen
Ueberblick ueber den hard- und softwaremaessigen Aufbau von Prozessrechnern; Abgrenzung zu wissenschaftlichen und kommerziellen Rechnern.

b) Maschinensprache
Funktionsweise eines Rechners; Ausfuehrung von Elementarbefehlen; Programmieren in Maschinensprache.

c) Assemblersprache
Befehlsarten- und Formate einer Assemblersprache; Macro- Technik; Progr. in Assemblersprache.

d) Hoehere Programmiersprachen
Allgemeines ueber hoehere Programmiersprachen; Uebersetzer; Binder; Anforderungen an eine hoehere Prozessrechner-Programmiersprache; Aufbau solcher Sprachen.

e) Prozess- FORTRAN und BASEX
Vorstellung dieser fuer die Prozessprogrammierung erweiterten technisch-wissenschaftlichen Sprachen.

f) PEARL
Entstehung von PEARL; Vorstellung der PEARL Sprachelemente.

Den Abschluss der Kapitel b) bis f) bildete jeweils der Unterpunkt "Wertung".

Der Schwerpunkt dieser Vorlesung lag damit auf der Vermittlung eines Sprachverstaendnisses fuer den Bereich der Prozessdatenverarbeitung. PEARL hatte dabei einen sehr hohen Stellenwert, da die Vorlesung quasi in der Vermittlung dieser Sprache gipfelte.

Da bei den Hoerern, die ueberwiegend aus Maschinenbau- und Kybernetikstudenten im Hauptstudium bestehen, keine oder nur einfache FORTRAN Programmierkenntnisse vorausgesetzt werden konnten, musste "ganz unten" angefangen werden. So wurden die heute wichtigen Problemkreise wie z.B. die Software- Entwurfstechniken und Hilfsmittel, das Prozesskonzept, die Synchronisation asynchroner Prozesse, die Softwarezuverlaessigkeit sowie Software fuer verteilte

Prozessautomatisierungssysteme nur am Rande oder gar nicht behandelt.

Zum Sommersemester 1981 wurde die Vorlesung voellig ueberarbeitet um den heute umfangreicheren DV Vorkenntnissen der Hoerer und den geaenderten Lehrinhalten Rechnung zu tragen. Die Vorlesung in der neuen Form gliedert sich wie folgt:

a) Grundlagen
Struktur von Prozess- und Microrechner; Grundlagen der Programmierung; Dokumentation; Prinzipien der Programmentwicklung; Echtzeit- und parallele Programmierung; Synchronisation; Deadlock;

b) Softwareentwurf
Phasen der Systemanalyse; Systemanalyse und Pflichtenheft; Systementwurf und Implementierung; das Entwurfshilfsmittel EPOS; Entwurfsbeispiel.

c) Die Echtzeitprogrammiersprache PEARL
Ueberblick ueber die PEARL Eigenschaften und Sprachelemente; Fallstudie.

d) Automatisierungssysteme
Software- Strukturierung mittels Bausteinsprachen; Software fuer Spezialrechner am Beispiel eines Prozessvideosystems.

Die Vorlesung in dieser Form besitzt damit drei Schwerpunkte:

1) ein sehr breites Grundlagenkapitel bis hin zum Prozesskonzept,

2) Systemanalyse und Softwareentwurf

und

3) PEARL als leistungsfaehige Prozessprogrammiersprache.

Zusaetzlich wurde ein straffer Uebungsbetrieb eingerichtet, der drei praktische Microrechner- (Assembler) und zwei praktische PEARL Uebungen umfasst. Die PEARL Uebungen sind "Prinzip des Anfahrens eines Dampferzeugers" und das Problem der "Dining Philosophers"(siehe unten). Beiden PEARL Uebungsaufgaben liegen Problemstellungen zugrunde zu deren Loesung parallele Prozesse (mehrere Tasks) erforderlich sind.

Bei der erstmaligen Durchfuehrung der Vorlesung im SS 1981 zeigte es sich, dass das Grundlagenkapitel noch etwas ueberladen ist. Die Lehrinhalte des Themenkreises "Echtzeit- und parallele Programmierung" muessen deren Komplexitaet wegen noch in ein eigenes Kapitel ausgelagert werden. Dadurch koennten evtl. auch Petri- Netze noch mit aufgenommen werden.

Der Aufbau, die fuer das Verstaendnis von asynchronen Prozessen grundlegenden Ideen und Mechanismen separat und vor der Einfuehrung eines speziellen Sprachkonzeptes zu behandeln, hat sich jedoch m.e. sehr bewaehrt. So kann z.B. bei der Einfuehrung der Sprache PEARL und bei den PEARL-Uebungen auf ein Grundverstaendnis dieser Thematik aufgebaut werden.

Die Uebungen am Mikrorechner- Kit und Prozessrechner tragen m.e. sehr zum Verstaendnis des sonst doch etwas trockenen Lehrstoffes bei. Programmierkenntnisse koennen dabei jedoch kaum vermittelt werden. Bei den PEARL- Uebungen wurden die (problemloesenden) PEARL- Programme fertig vorgegegeben. In den Uebungsstunden wurden die Programme dann gruppenweise getestet und dabei spezielle (versteckte) Probleme diskutiert. In der Uebungsaufgabe "Anfahren eines Dampferzeugers" waren dies zur Laufzeit auftretende PEARL Signale, bei den "Dining Philosophers" war dies die Deadlock- und Starvation- Problematik bei der Verwendung von Semaphoren.

3. PEARL VERSUCHE IM HAUPTFACHPRAKTIKUM

Bei allen Versuchen im Hauptfachpraktikum (fuer Maschinenbau- Studenten mit dem Hauptfach "Prozessdatenverarbeitung") stehen spezielle technische Problemstellungen und deren prinzipielle Loesung im Vordergrund. So wird auch bei den beiden PEARL-Praktikumsversuchen "Prozessanfahrsteuerung" und der "Kontinuierlichen Abwasserneutralisation im Durchfluss" nicht speziell auf die Sprache PEARL eingegangen. PEARL wird als bekannt vorausgesetzt und sofort als Handwerkszeug fuer die Formulierung der Problemloesung benutzt.

Bei dem Versuch "Prozessanfahrsteuerung" wird die Problemloesung auch alternativ in der aus BASIC abgeleiteten, interpreterunterstuetzten Prozessprogrammiersprache BASEX formuliert. Dadurch soll dem Studenten der Vorteil problemgerechterer Formulierung in PEARL sowie das schnellere Arbeiten mit einer Interpretersprache nahegebracht werden.

4. DER PEARL- KURS VOM 14.-18. SEPT.1981

Der in Zusammenarbeit mit dem VDI- Bildungswerk und dem PEARL Verein durchgefuehrte Kurs "Programmieren mit PEARL" entsprach nach Konzept und Kursunterlage dem an der TU Berlin erarbeiteten Kompaktseminar "Systematisches Programmieren mit PEARL". Da ueber die didaktischen und methodischen Prinzipien dieses Seminares im Vortrag der Herren Nagel und Nebel berichtet wird, moechte ich meine Ausfuehrungen auf die Durchfuehrung des Uebungsbetriebes und die Kursbeurteilung durch die Teilnehmer beschraenken.

DER UEBUNGSBETRIEB

Die praktischen Uebungen, die etwa 40% der zur Verfuegung stehenden Zeit in Anspruch nahmen, wurden gruppenweise am Prozessrechner PDP 11/34 des Institutes durchgefuehrt. Dazu wurde ein Uebungsraum mit drei Datensichtgeraeten eingerichtet. Bei 11 Kursteilnehmer ergaben sich damit drei Gruppen mit 2 x vier und 1 x drei Teilnehmern.

Die Uebungsgruppenen wurden waerend des Kurses staendig von drei Uebungsleitern

betreut. Beim Kursbeginn wurde zudem noch eine Kurzbeschreibung der wichtigsten Befehle am Datensichtgeraet ausgegeben (Editor- Befehle, Aufruf des PEARL Compilers, des Binders, wichtige Betriebssystembefehle).

Ablauf

Die erste praktische Uebung war die Eingabe des im ersten Kapitel des Skriptes fertig vorgegebenen PEARL Programmes. Dies diente in erster Linie dazu, die Teilnehmer mit dem Rechner bzw. der fuer die PEARL Programmentwicklung notwendigen Werkzeuge vertraut zu machen.

Die zweite Uebung war die Vervollstaendigung des im zweiten Kapitel des Skriptes angegebenen PEARL Programmes "Sortieren einer Fahrerdatei", um eine vernuenftige Ausgabe der sortierten Datensaetze zu erreichen.

Als dritte Uebung war die Programmierung eines einfachen Autorennens vorgesehen (pro Auto eine Task, Rundenzeiten durch Zufallszahlen ermittelt). Diese Uebung wurde von uns durch die Uebung der "Dining Philosophers" ersetzt.

Da dieser Uebung eine besonders anschauliche Problemstellung zugrunde liegt und sie auf einfache Weise relativ viel zeigt, ist diese Uebung im folgenden kurz skiziert.

Die Uebung "Dining Philosophers"

Die Aufgabenstellung der Dining Philosophers stammt urspruenglich von E.W. Dijkstra und lautet:

> Fuenf Philosophen sitzen an einem runden Tisch. Die Philosophen verwenden ihre Zeit entweder zum Denken oder zum Essen. Als Essen steht ein kompliziertes Spaghettigericht auf dem Tisch, dessen Verspeisung nur mit Hilfe zweier Gabeln moeglich ist. Zur Verfuegung stehen nun gemaess Bild 1 fuenf Gabeln. Hat ein Philosoph Hunger, so greift er zuerst zu seiner linken Gabel, dann nach seiner rechten Gabel. Andere Gabeln sind ihm nicht zugaenglich.

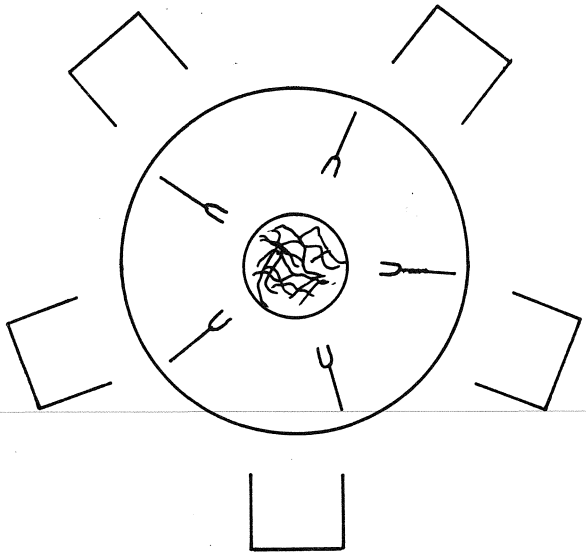


Bild 1: Dining Philosophers

Diese Aufgabenstellung wurde fuer die Uebung etwas abgewandelt:

> Die fuenf Gabeln sind nicht gemaess Bild 1 reihum aufgelegt, sondern werden aus einem zentralen Geschirrbehaelter entnommen bzw. dorthin zurueckgelegt.

PEARL Formulierung:

Die Umsetzung der Aufgabenstellung in ein PEARL- Programm ist sehr einfach. Es sind folgende Abbildungen vorzunehmen:

- Das Verhalten jedes Philosophen wird durch eine Task realisiert. Alle Philosophen besitzen dasselbe Verhalten. Fuer Philosoph 1 ergibt sich damit:

```
Phil1: TASK;
   DCL Name FIXED INIT(1);
   REPEAT
      CALL AKTION(Name);
   END;
END;
```

- Zum Systemstart wird zusaetzlich eine Starttask eingebaut.

- Das Greifen und Zuruecklegen der Gabeln wird durch REQ und REL auf eine Semaphore GABEL simuliert. Die Semaphore Gabel wird bei ihrer Deklaration mit 5, entsprechend den fuenf Gabeln, initialisiert.

```
DCL GABEL SEMA PRESET (5);
```

- Zur Beobachtung des laufenden Systems werden zur Laufzeit bestimmte Systeminformationen auf dem Terminal ausgegeben. Besonders interessant sind dabei die Uhrzeiten fuer die auftretenden Ereignisse (Systemroutine NOW).

  Fuer die Prozedur AKTION ergibt sich damit in etwa folgender Aufbau:

```
AKTION: PROCEDURE( NAME FIXED);
   PUT NOW,' Phil',NAME,' denkt'
       TO BILDSCHIRM BY ...;
   AFTER 10 SEC RESUME;

   REQUEST GABEL;
   PUT NOW,' Phil',NAME,' eine Gabel'
       TO BILDSCHIRM BY ...;
 ( AFTER 20 SEC RESUME; )
   REQUEST GABEL;

   PUT NOW,' Phil',NAME,' zwei Gabeln'
       TO BILDSCHIRM BY ...;
   / 10 Sec Essen /
   AFTER 10 SEC RESUME;

   RELEASE GABEL;
   RELEASE GABEL;
END; / Proc. AKTION /
```

Anm.: Das eingeklammerte Statement ist einzufuegen, um die in der Problemstellung versteckte moegliche Systemverklemmung wahrscheinlicher zu machen. Die Systemverklemmung tritt genau dann auf, wenn jeder Philosoph genau eine Gabel besitzt.

DIE KURSBEURTEILUNG

Den Kursunterlagen war ein vorgedruckter Kursbeurteilungsbogen des VDI- Bildungswerkes beigefuegt, der nach Ende des einwoechigen Kurses abgegeben werden konnte.

Der Kursbeurteilungsbogen bestand aus einem formalen Teil, in dem Aussagen durch Ankreuzen vorgegebener Formulierungen gemacht werden konnten, und einem Teil fuer freie Formulierungen bzw. Anregungen.

Im formalen Teil war anzukreuzen:

- Gesamturteil
- Empfehlung fuer Fachkollegen

fuer die einzelnen Kurstage (Dozenten):

- Inhalt
- Anforderungen
- Vortragsweise

Trotz des etwas kleinen Mengenrahmens von nur 11 Teilnehmer haben wir die abgegebenen Kursbeurteilungsbogen ausgewertet. Danach ergibt sich als

- Gesamturteil (Bild 2)	"sehr gut"	11%
	"gut"	89%
	"mittelmaessig"	--
	"schlecht"	--
- Empfehlung fuer Fachkollegen		
	"empfehlenswert"	75%
	"teilweise empfehlenswert"	25%
	"wenig geeignet"	--
- Einzelbeurteilungen (Summe ueber alle Kurstage)		
Inhalt	"ausgewogen"	87%
	"Teile zu speziell"	3%
	"Teile zu allgemein"	5%
	"Teile zu speziell"	5%
Anforderungen	"richtig"	87%
	"zu niedrig"	5%
	"zu hoch"	8%
Vortragsweise		
	"gut"	70%
	"manches zu knapp"	3%
	"Thema ueberfluessig"	3%
	"Teile zu breit"	12%
	"Teile zu gedraengt"	11%

Zur Beurteilung der Einzeltage siehe Bild 3. Die in Bild 3 dargestellten Balken kommen durch die Gewichtung der Einzelaussagen nach ihrer Guete zustande (gute Aussagen x 2).

4. ERFAHRUNGEN

Ueberdenkt man die im Rahmen der verschiedenen PEARL Lehraktivitaeten gewonnenen Erfahrungen, so zeichnen sich folgende Ergebnisse ab:

(1) Der Stellenwert von praktischen Uebungen beim Erlernen von PEARL ist sehr hoch.

(2) Es genuegt nicht, nur die einzelnen PEARL Sprachelemente zu lehren. Es muss zusaetzlich ein Verstaendnis der zugrundeliegenden Ideen besonders des Taskings, der Synchronisation und des Dation- Konzeptes vermittelt werden.

(3) Bei praktischen Uebungen moeglichst kleine Uebungsgruppen (max. drei Teilnehmer).

(4) Neben der Vermittlung der Sprache PEARL ist die Vermittlung einer sauberen Programmiertechnik (Top- Down Entwurf, strukturierte Programmierung) sehr wichtig.

(5) Das Einueben, bzw. Kennenlernen der einzelnen PEARL Sprachelemente an Fallstudien bzw. an Beispielen hat sich sehr bewaert.

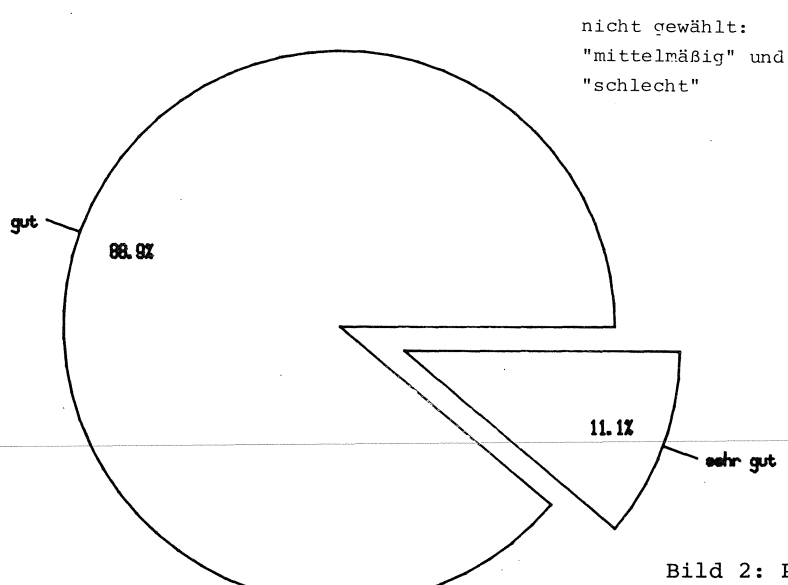


Bild 2: PEARL Kurs 14.-18. 9. 81 Beurteilung Gesamturteil

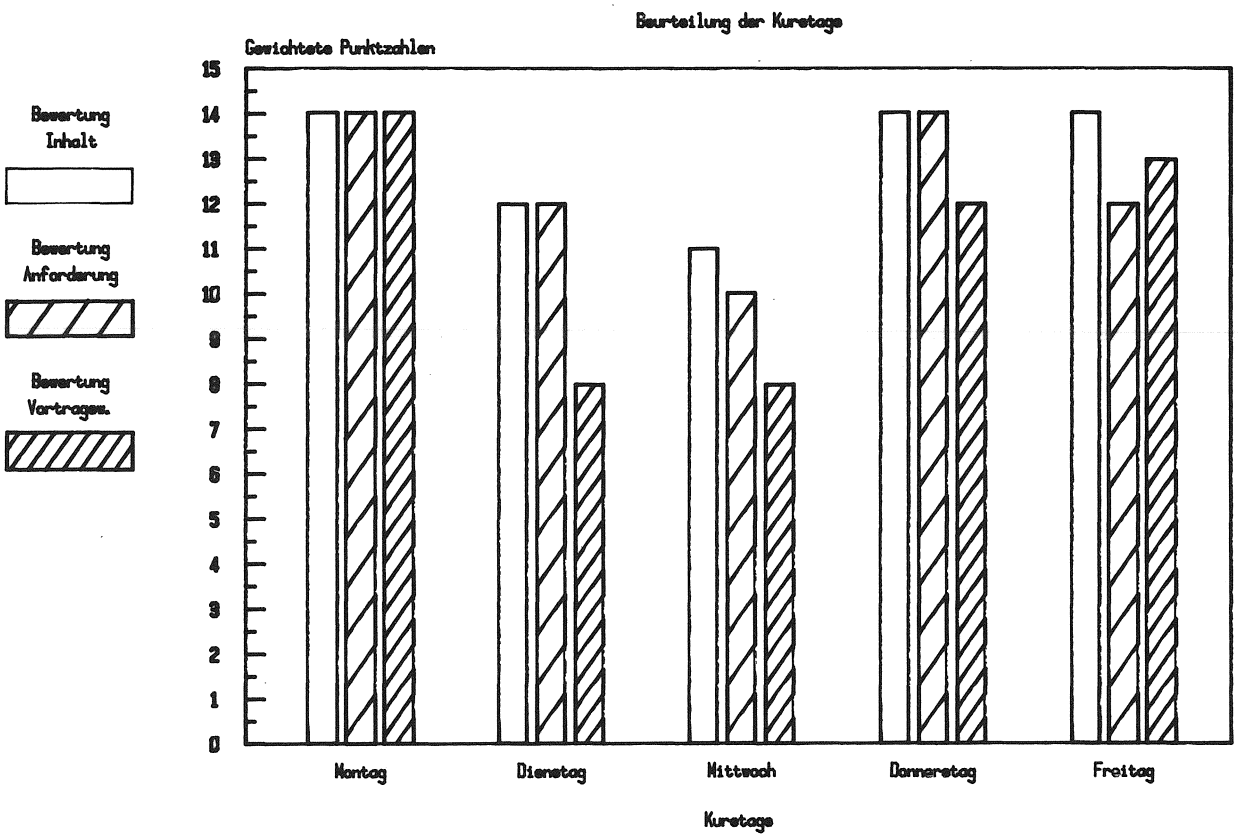


Bild 3: Beurteilung PEARL Kurse 14.-18. 9. 81

ANSCHRIFT DES AUTORS

Roehrich, Thomas

Institut fuer Verfahrenstechnik
und Dampfkesselwesen
Abteilung fuer Stromerzeugung
und Automatisierungstechnik
Pfaffenwaldring 9
7000 Stuttgart 80

Tel. 0711/784-6203

Überblick über Lehr- und Ausbildungsveranstaltungen mit PEARL

Institution	Welche Lehr- oder Ausbildungsveranstaltungen mit/über PEARL werden durchgeführt?	In welchem Semester? (bei Vorlesungen/ Übungen)	Seit wann?	Zahl der Hörer bzw. Teilnehmer?	Gesamtzahl der bisher in PEARL Ausgebildeten?
Universität Stuttgart Lehrstuhl für Regelungstechnik u. Prozeßautomat.	Prozeßautomatisierung I Fachpraktikum Prozeßautomatisierung	6. Semester 7. Semester	1975 1976	ca. 100 ca. 60	700 300
Institut für Rundfunktechnik, München	Anwenderkurse für PEARL		1981	ca. 20	80
Hochschule der Bundeswehr, München, FB Informatik	Prozeßrechensystem Übungen hierzu (im Aufbau)	5. Trimester 5. Trimester	1978 --	ca. 40 ca. 40	180
FH Konstanz, FB Informatik	Prozeßrechnersystem	8. Semester (=Pflichtfach f. 2 Studieng.)	PEARL SS 81 FORTRAN SS 72	ca. 20	ca. 20
Gesellschaft für Reaktorsicherheit	Anwenderkurs (auf Englisch)		1981	12	12
Universität Stuttgart Inst.f.Verfahrenstechnik u.Dampfkesselwesen Fakultät Maschinenwesen	Vorl. Prozeßrechnersprachen u. Programmierkurse Hauptfachpraktikumsversuche	5. - 8. Sem. Praktiker 5. - 8. Sem.	1976 1979 1976	ca. 40 je 12 ca. 20	300 20 150
Elektr. Steuerung u. Regelung, Universität Bochum	keine	--	--	--	--
FH Bielefeld FB Elektrotechnik	Automatis. DV Prozeß-DV Spez. Progr.Spr. + Praktika	2. u. 3. Sem. 4. u. 5. Sem. 6. Sem.	1980 1980 1980	ca. 60 ca. 60 ca. 5	ca. 300
Digital Equipment GmbH Schulungszentrum	PEARL-Sprachkurs PEARL unter RSX-11M		1980	max. 16 max. 16	ca. 15 ca. 50
Universität (GH) Paderborn	Übungen und Vorlesungen PEARL I/II	5./6.Semester oder höher	1978	15 - 25	ca. 70
FH Furtwangen	Vorlesung: Prozeßinformatik + Praktika	7. Semester	1981	10 - 20	ca. 20
TU Berlin-FSP/PV Prozeßrechnerverbund	1. Schulung f. Anwender "Syst.Progr.m.PEARL" 2. PEARL-Kompaktkurs "Syst.Progr.m.PEARL" auch Schulung des PEARL-Vereins/VDI	Forschung	1979 1980 1981	10 - 20 je 18 11 - 18	50 36 50
Siemens: Schule für Prozeßrechnertechnik	PEARL-Sprachkurs		1978	max. 20	ca. 90
FH Hagen, FB Elektrotechn.	keine	--	--	--	--
Universität Erlangen-Nürnberg	PEARL-Vorlesung Prozeßrechnerpraktikum	ab 4. Sem.	1975	10 - 15	ca. 75
FH Gießen-Friedberg Bereich Gießen	Vorlesung über PEARL in zwei Teilen	ab 4. Sem.	1980	ca. 15	15

Gesamtzahl der Lehrstunden für PEARL pro Lehrveranstalt. (einschl. Übungen)	Bei Ausbildungskursen: Wie viele Kurse/Jahr?	Welche Vorkenntnisse werden gefordert?	Welche Prozeßrechner und welches PEARL-System wird verwendet?	Bemerkungen
ca. 10 6			AEG 60-50, ASME-PEARL später: AEG 80-20 AEG-PEARL	
40	4 pro Jahr	keine Vorkenntnisse	HP 3000, HP 1000	
ca. 6 ca. 20		allg. Programm.Kenntn. Grundlagen Betriebssyst.	PERKIN ELMER (32-Bit-Form) eigene Komp.	
ca. 48 in Studentengruppen	in SS und WS gleich	Vorlesung u. Übungen Einführung Betriebssysteme	PDP - 11/40	
30	2/1981	für Programmierer	Norsk Data	
ca. 10 6	etwa 2	Programmierkenntnisse DV-Grundlagen	Siemens 330 AEG 80-20 KAE EPR 1300 PDP 11/34	
--	--	--	HP 2100, HP 1000 kein PEARL	
ca. 70 ca. 70 ca. 4			Krupp-Atlas EPR 1300	
32/ 5 Tage 64/10 Tage	2 2	höhere Programmierspr. PEARL-Erfahrung	11/44 11/44	Theorie u. Praxis je Hälfte d.Stunden
ca. 60		Grundlagen des Programmierens	BBC DP 1000 (PDP 11/35)	
5		Progr.Sprachen, Betr.-Systeme, Progr.Technik	PDP 11/34 RSX - 11 M	
60 40 UE 40 UE	nach Bedarf ca. 2 - 3 ca. 4	keine keine	HP 3000 HP 1000 HP 3000 PDP 11/34 (IVD)	auch typisch Einzelschulung am Projekt
10 Tage	ca. 3	Grundlagen der Program.	R 30/330 PEARL 300	
--	--	--	DIETZ 621/8 ohne PEARL	
ca. 70 ca. 70	1 1	eine höhere Programmiersprache	R 30/PEARL 300	
ca.90 I:2hV + 2hÜ II:1hV + 1hÜ	1 Kurs: PEARL I 1 Kurs: PEARL II		PDP - 11/60	

Presseschau

Handelsblatt 23.9.1981

Prozeßsteuerung

Echtzeitprogrammieren in „höheren" Sprachen

Von ANGELIKA LOEWENHEIM

HANDELSBLATT - TL, 22. 9. 1981

Wo Computer technische Prozesse überwachen, steuern und regeln, stellen Sicherheit und Zuverlässigkeit hohe Ansprüche. Das gilt gleichermaßen für die Geräte (Hardware) wie für die Programme (Software). Während bei der Hardware z.B. verteilte Multiprozessoren die Systemverfügbarkeit erhöhten, hinkt der Software-Bereich noch hinterher. Höhere Sprachen wie Pearl könnten hier weiterhelfen.

Die Computer Zeitung 19.10.1981

PEARL für den Computereinsatz im öffentlichen Nahverkehr

PEARL (Process and Experiment Automation Realtime Language) – eine in Deutschland entwickelte, vielfältig einsetzbare höhere Programmiersprache für Prozeßrechner – erschließt sich nach und nach komplette Branchen, so z. B. den öffentlichen Nahverkehr. In einem Pionierprojekt bei der ÜSTRA Hannoversche Verkehrsbetriebe AG wird mit Förderung des Bundesministers für Forschung und Technologie ein standardisiertes Betriebsleitsystem für den öffentlichen Nahverkehr mit umfangreicher PEARL-Software unter der Bezeichnung BON entwickelt.

Erlanger Nachrichten 07.10.1981

Steuerung technischer Prozesse per Computer

Experten-Tagung diskutierte Probleme

Schickt ein Computer eine Mahnung zu einer längst bezahlten Rechnung, so ist das für den Absender zwar peinlich, aber kein Beinbruch. Fatale Folgen können solche Programmierfehler aber bei der Steuerung technischer Prozesse haben. Man kann dabei an Großanlagen (wie Kraftwerke) denken oder auch an ein mikroprozessorgesteuertes Antiblockiersystem im Auto. Wie solche — unter Umständen lebensgefährliche Pannen vermieden werden können, war das Thema einer Tagung in Erlangen.

Ursache der Fehler ist oft, daß sich der Programmierer auf Details des Computers konzentrieren muß. In vielen Anwendungsbereichen konnte in den letzten 15 Jahren die Programmqualität durch die Verwendung sogenannter problemorientierter Programmiersprachen wesentlich verbessert werden. Diese erlauben dem Anwender, sich auf eine saubere Lösung des Problems zu konzentrieren und nicht auf die Eigenarten des Computers.

Für die Steuerung technischer Prozesse ist in Deutschland mit Förderung der Bundesregierung eine solche Programmiersprache unter dem Namen PEARL entwickelt worden. An der Technischen Fakultät der Universität Erlangen-Nürnberg (Lehrstuhl für Programmiersprachen, Prof. Dr. Schneider) haben sich nun 50 Vertreter von Universitäten, Fachhochschulen und verschiedenen industriellen Ausbildungseinrichtungen getroffen und diskutiert, wie angehende und bereits im Berufsleben stehende Ingenieure in dieser neuen Technik geschult werden können.

Neben didaktischen Gesichtspunkten, Lernzielkatalogen und Ausbildunskapazitäten kamen auch erste Erfahrungen zur Sprache. Besonders konnte hervorgehoben werden, daß eine solche Sprache geeignet ist, Studenten mit Beispielen aus der Praxis zu konfrontieren.

Computertechnik 30.10.1981

Echtzeit-Programmiersprache Pearl senkt Softwarekosten in der Prozeßautomation

Hoher Wert auf Zuverlässigkeit gelegt

München (wp) — Die Softwarekosten nehmen einen immer größer werdenden Anteil an den gesamten Projektkosten ein. Daneben wird für die kommenden Jahre ein erhebliches Defizit an Programmierern vorausgesagt. Die Auswahl der richtigen Programmiersprache hat daher bei Neuentwicklungen einen sehr hohen Stellenwert bekommen, da hier durch eine Fehlentscheidung immense Mehraufwendungen die Folge sein können. Bei Anwendungen in der Prozeßautomatisierung erhebt die noch relativ junge Echtzeit-Programmiersprache Pearl den Anspruch für sich, durch leichte Erlernbarkeit und eine klare Struktur dem unbedarften Prozeßingenieur das Selbstprogrammieren zu ermöglichen und dadurch die Softwarekosten zu senken.

Die Computer Zeitung 19.10.1981

Softwarekosten werden minimiert – Aufwand wird reduziert

Deutsche Realzeit-Programmiersprache

Die höhere, anwender- und problemorientierte Prozeßrechnersprache PEARL (Process and Experiment Automation Realtime Language) wurde – ausgehend von den Bedürfnissen mehrerer Industrieunternehmen und Hochschulinstitute – unter Förderung des BMFT Anfang der siebziger Jahre in der Bundesrepublik entwickelt. Ausgangspunkt war der Wunsch nach einer einfach erlernbaren, komfortablen Sprache, die es dem Prozeßingenieur selbst ermöglicht, seine Automatisierungsaufgaben zu lösen, ohne auf DV-Spezialisten angewiesen zu sein. Damit sollten die Entwicklungskosten für neue Software minimiert und der Aufwand für Wartung und Aktualisierung bestehender Programme reduziert werden.

Die Computer Zeitung 19.10.1981

Die Entwicklung, Förderung und Standardisierung der Programmiersprache PEARL

Die Entwicklung der deutschen Realzeit-Prozeßrechnersprache PEARL (Process and Experiment Automation Realtime Language) nahm Ausgang im Jahre 1969 im Rahmen der Studiengruppe Nuklearelektronik des BMBW. Vorausgegangen waren verschiedene Ansätze, und zwar bei der GFK Karlsruhe, an der Universität Erlangen, bei BBC und Siemens, eine neue Prozeßautomatisierungssprache zu entwickeln, weil die vorhandenen höheren Sprachen nicht den spezifischen Anforderungen entsprachen. Diese getrennt laufenden Entwicklungen konnten koordiniert werden, zahlreiche weitere Firmen kamen in den Jahren 1970/71 hinzu und stellten ihrerseits entsprechende Förderungsanträge, so daß die Betreuung der Aktivitäten zusammengeführt werden mußte und 1972 vom BMBW an das Projekt PDV das BMFT überging. Ende 1972 lag auf gemeinsamer Basis die erste komplette Beschreibung von PEARL vor, die 1973 als PDV-Bericht veröffentlicht wurde. Die Unterstützung durch das BMFT erfolgte bis zum Jahre 1979. Insgesamt wurden 30 Mio DM aufgewendet. Die Einheitlichkeit von PEARL wird durch die Normung gesichert. Seit 1978 liegt der Normentwurf DIN 66253 Teil 1 „Basic PEARL" vor. Er ist im Frühjahr 1981 als Vornorm herausgekommen. Im November 1980 ist der Normentwurf DIN 66253 Teil 2 „Full PEARL" erschienen. Parallel dazu wurde PEARL zur internationalen Normung bei der ISO eingereicht.

Die Computer Zeitung 19.10.1981

Siemens-System 300 erstmals in der Netzleitstelle Deggendorf

Im Dialog mit dem Prozeßrechner

PEARL 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 höhere Zuverlässigkeit aus. Siemens-Systeme 300 wurden erstmals im Projekt Netzleitstelle Deggendorf der Energieversorgung Ostbayern AG (OBAG) eingesetzt. Die OBAG ist ein regionales Elektrizitätsversorgungsunternehmen. Auf einer Fläche von 22 000 qkm versorgt sie ein Drittel von Bayern mit elektrischer Energie.

PEARL im In- und Ausland aktiv

Daß PEARL (Process and Experiment Automation Realtime Language) – die deutsche Programmiersprache für Prozeßrechner – nicht nur hierzulande auf großes Interesse stößt, sondern auch jenseits des Atlantiks in den USA und Südamerika, zeigt der ständig wachsende Anwenderkreis.

Um die Nachfrage nach PEARL in den Anwenderkreisen und das entsprechende Angebot von Herstellerseite weiter zu steigern, werden in nächster Zukunft zahlreiche Veranstaltungen durchgeführt, u.a. ein Programmierkurs vom VDI-Bildungswerk (14.–19.9. in Stuttgart), eine Präsentation vor italienischen Prozeßingenieuren (28.–30.9. in Cernobbio/Mailand), eine Fachtagung mit Workshop für PEARL-Lehrende (1.–2.10. an der Universität Erlangen) sowie ein Informations-Seminar in Philadelphia/USA (29.3.–2.4.82).

Außerdem sind Mitglieder des PEARL-Vereins als Referenten auf bedeutenden Fachtagungen wie BIAS 81 – Control of Industrial Processes (6.–7.10. in Mailand) und 3. DV-Konferenz der UNIPEDE (6.–8.10. in Zürich) vertreten.

Auf der Systems (19.–23.10. in München) stellt der PEARL-Verein auf eigenem Stand aus (Halle 7, Nr. 7301).

Die Computer Zeitung 19.10.1981

Was unterscheidet PEARL von anderen Programmiersprachen?

Für Prozeßrechner-Anwendungen

PEARL (Process and Experiment Automation Realtime Language) wurde als Werkzeug für den Anwendungsingenieur in der Prozeßautomation entwickelt. Es unterscheidet sich damit wesentlich von allen anderen Lösungen in diesem Bereich.

f + h fördern und heben Sept. 1981

Leserbriefe

Sehr geehrter Herr Dr. Elzer,

die Herren L. Frevert und R. Rössler haben im Band 2, Juni 1981 der PEARL-Rundschau einen Sprachvergleich von PEARL und Concurrent PASCAL in Dialogform vorgenommen. Da mir die Adressen dieser Herren nicht vorliegen, bitte ich Sie, stellvertretend den Autoren meinen Dank für die erfrischende Form dieses Sprachenvergleichs auszusprechen.
Es zeigt sich, daß auch schwierige techn. Aussagen "unterhaltsam" darstellbar sind.
Bei dieser Gelegenheit möchte ich folgende Anregungen zur PEARL-Rundschau geben:

- jeweils fortlaufende Seitennumerierung über alle Hefte des Jahres
- am Jahresende ein getrenntes Inhaltsverzeichnis über alle Beiträge des Jahres (da beim Binden der Hefte der Einband entfällt)
- ggf. gegen Aufpreis ein Bucheinband oder Buchdeckel für die erschienenen Jahreshefte mit dem PEARL-Aufdruck und Jahrgangsangabe.

Mit freundlichen Grüssen

Ing. grad. H. Räuber
c/o Badenwerk AG