PEARL Rundschau

Inhalt

Vorwort der Schriftleitung 137

G. Fickenscher
Standardisierte Programmiersprachen in der Bundeswehr 139

H. Weber
SASA-Schießausbildungs-, Sicherheits- und Auswerte-Anlage . . . 143

D. Krönig
ARES-Artillerie Raketen Einsatzsystem 151

Aberl/D. Fenzel
PEARL-System für ein athmospärisches Meß- und Auswertesystem ATMAS 157

H. Steusloff
Mehrrechner-PEARL für Mehrrechner und Mehrprozessor-Systeme . 159

November 1982 Band 3 Nr. 4

PEARL Rundschau

November 1982
Band 3
Nr. 4

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 in 7000 Stuttgart 1, Seidenstraße 36

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.

Vorwort der Schriftleitung

Die Tagung "PEARL in der Wehrtechnik", die am 6.10.82 in Koblenz stattfand, ordnet sich ein in eine Serie von Veranstaltungen, die von staatlichen Behörden mit technischen Aufgaben zur Information ihrer Mitarbeiter über Eigenschaften und Vorzüge von PEARL abgehalten werden. Sie kann jedoch als besonders erfolgreich bezeichnet werden, da sie mit über 80 Teilnehmern aus dem BMVg und dem BWB und ausgiebigen Diskussionen das große Interesse an PEARL demonstrierte, das in diesen Behörden besteht. Es ist deshalb auch geplant, im Frühjahr 1983 die Veranstaltung für die "Bedarfsträger", also Angehörige der Truppenteile, zu wiederholen.

Der Erfolg einer Tagung ist natürlich immer in hohem Maße von ihren Organisatoren abhängig. Deshalb soll an dieser Stelle besonders dem technischen Organisator, Herrn Fickenscher vom BWB, und dem Tagungsleiter, Herrn Dr. Munck vom BMVg, im Namen des PEARL-Vereins für ihren großen Einsatz gedankt werden, mit dem sie dieser Veranstaltung zum Gelingen verholfen haben.

Höheren Programmiersprachen kommt ja im wehrtechnischen Bereich eine besondere Schlüsselstellung zu. Zum einen sind die Anwendungen meist so komplex, daß die Verwendung von Assembler sich schon lange von selbst verbietet. Zum anderen ist wegen des hohen Anteils an Programmpflege und -wartung, die außerdem infolge der "systembedingten" Personalfluktuation unter erschwerten Bedingungen stattfindet, eine Standardisierung auf wenige Programmiersprachen besonders dringlich.

Umso erfreulicher ist es, daß dies nun - nach Großbritannien, Frankreich und den USA - auch in der BRD gelungen ist. Noch erfreulicher ist, daß PEARL zu diesen Standardsprachen gehört, nachdem ja sonst meist "der Prophet im eigenen Vaterlande nichts gilt". Die bittere Wahrheit dieses Satzes mußten besonders die Entwickler von PEARL wieder und wieder erkennen. So mutet es wie ein makabrer Scherz an, daß dem PEARL-Verein immer noch vom Finanzamt die Gemeinnützigkeit verweigert wird, wenn man weiß, wieviel ehrenamtlichen Aufwand und Freizeit gerade die "Aktivisten" dieser Organisation investieren. Vielleicht wäre es einmal eine Aufgabe für Inhaber "höherer Stellen", die sich inzwischen von der Notwendigkeit und den Qualitäten einer der wenigen erfolgreichen nationalen Entwicklungen auf dem DV-Sektor überzeugen konnten, unter ihren Kollegen von der geldeinnehmenden Seite etwas aufklärend zu wirken.

Der geschätzte Leser möge sich aber durch diese etwas kritischen Bemerkungen nicht davon abhalten lassen, den interessanten technischen Inhalt dieses Heftes zu genießen und zu würdigen.

Für die Schriftleitung
gez. Dr. P. Elzer

Standardisierte Programmiersprache in der Bundeswehr

von **G. Fickenscher**, Koblenz

Zusammenfassung: Die elektronische Datenverarbeitung gewinnt innerhalb der Bundeswehr immer mehr an Bedeutung, sei es als DV-Anteil in Wehrmaterial, sei es als rechnergestütztes Führungsinformationssystem. Damit die Kosten für Erstellung, Pflege und Änderung dieser Software-Systeme nicht ins Uferlose wachsen, ist die Bundeswehr gezwungen, sich auf wenige ausgewählte Programmiersprachen mit dafür geeigneten Programmierumgebungen zu beschränken. Nach einem längeren Prozeß wurden PEARL und Ada als die Standard-Programmiersprachen bestimmt. Das zugehörige Programmiersystem ist SPERBER (Standardisiertes Programm-Erstellungssystem für den Rüstungsbereich).

Abstract: In the German Armed Forces electronic data processing becomes more and more important, either as part of armament equipment, either as command, control and information system. To reduce the costs for development and maintenance of these software systems the German Armed Forces have to reduce the variety of the used programming languages to few and proper programming languages with related programming support environments. After a selection process it was decided to have PEARL and Ada as the standard programming languages. The programming support environment is SPERBER (Standardized Programming Support Environment for the Armament Sector).

1. Einleitung

In der Bundeswehr (Bw) wird elektronische Datenverarbeitung in zwei großen Bereichen eingesetzt:

- dem administrativen Bereich (Personalangelegenheiten, Haushaltsplanung, etc.)
- dem wehrtechnischen Bereich (Waffensysteme, Erprobungen, etc.)

Der administrative Bereich wird bei den weiteren Betrachtungen außer acht gelassen. Den wehrtechnischen Bereich kann man weiter untergliedern in

- technisch-wissenschaftliche Anwendungen
- DV-Anteile in Wehrmaterial
- rechnergestützte Führungsinformationssysteme

Bei den technisch-wissenschaftlichen Anwendungen kann die Vormachtstellung der Programmiersprache FORTRAN kaum angefochten werden, allein schon wegen der Fülle der vorhandenen Bibliotheken. Die Standardisierungsbemühungen innerhalb der Bw werden daher zunächst auf

- DV-Anteile in Waffensystemen und
- rechnergestützte Führungsinformationssysteme

beschränkt.

Bei diesen beiden Gebieten wird eine Palette von Rechnern - vom Spezialrechner ohne vorhandenes Betriebssystem bis zum Großrechner mit komfortablen Betriebssystem - eingesetzt. Die Anforderungen reichen von harter Realzeitdatenverarbeitung (z.B. Avionik-System) bis zu Stapelverarbeitung. Für die weitere Betrachtung ist es daher notwendig, die verwendeten Software-Systeme in Schichten zu zergliedern. Dabei genügt es jedoch, folgende Ebenen zu unterscheiden:

- Anwender-Software

  Diese Ebene realisiert das gestellte Problem (z. B. Feuerleitalgorithmus).

- Anwender-Applikations-Software

  In dieser Ebene sind weitgehend anwendungsneutrale Software-Pakete angesiedelt, die jedoch von dem jeweiligen Anwender parametrisiert werden müssen (z. B. Datenbanksystem).

- System-Applikations-Software

  Diese Ebene enthält weitgehend system-(rechner-)unabhängige Software-Pakete (z. B. Kompiliersysteme).

- System-Software

Jede dieser Ebenen stellt an die zu verwendende Programmiersprache unterschiedliche Anforderungen. Hinzu kommen die unterschiedlichen Anforderungen unterschiedlicher Anwendungsgebiete. Bei einer Standardisierung von Programmiersprachen muß dies berücksichtigt werden.

2. Historie

In den Jahren 1975/76 mußten Entscheidungshilfen gefunden werden, welche Programmiersprachen bei der Realisierung der rechnergestützten Führungsinformationssysteme, insbesondere EIFEL, und bei der Programmierung des Hauptrechners des Kampfflugzeuges MRCA/Tornado eingesetzt werden sollen. In beiden Fällen sollten höhere Programmiersprachen verwendet werden.

Wegen der allgemeineren Ergebnisse werden die Untersuchungen für die rechnergestützten Führungsinformationssysteme zuerst dargestellt.

2.1 Ada

Die Untersuchenden standen zu Beginn einer Fülle von Kandidatensprachen gegenüber. Es war daher notwendig, einen Kriterienkatalog zur objektiven Bewertung höherer Programmiersprachen zu erstellen. Dieser Katalog wurde 1978 veröffentlicht. Parallel dazu entstand Ende 1977 ein Kurzvergleich der Programmiersprachen CMS-2, CORAL 66, JOVIAL J3, LTR, PEARL (Avionic Subset) und SPL-3, in dem PEARL am besten abschnitt.

Um die Situation bzgl. Programmiersprachen innerhalb der Bw festzustellen, wurden Rechenzentren befragt. Das Ergebnis kann folgendermaßen zusammengefaßt werden.

Die verwendeten Sprachen sind:

50 %	Assembler-Sprachen (16 Dialekte)
25 %	FORTRAN (10 Dialekte)
10 %	COBOL
5 %	PL/1
10 %	ALGOL, ATLAS, JOVIAL, CMS-2 und andere

Die Programmiertätigkeit verteilt sich auf folgende Aufgaben:

20 %	Pflege
36 %	Änderung
32 %	Entwicklung
12 %	Erstellen von Hilfssoftware

Die Ergebnisse decken sich mit Ergebnissen ähnlicher Untersuchungen aus den USA. Die schiefe Verteilung der Programmiertätigkeiten ergibt sich aus der Verwendung von Assembler-Sprachen, Erstellung nicht wiederverwendungsfähiger Software und uneinheitlichen Programmiersystemen.

Im Rahmen der Untersuchungen stieß man auf das HOL-Programm des US-Verteidigungsministeriums (Ergebnis: Ada) und beschloß, daran mitzuarbeiten.

Nach der Definition von Ada wurde im deutschen Verteidigungsbereich das Vorhaben SPERBER (Standardisiertes Programm-Erstellungssystem für den Rüstungsbereich) ins Leben gerufen, damit Deutschland nicht von US-Entwicklungen abhängig ist.

SPERBER ist ein Pool von Tools zum ingenieurmäßigen Entwickeln von Software-Systemen in den Programmiersprachen Ada und PEARL für die Bundeswehr. Es wird dabei davon ausgegangen, daß grundsätzlich auf einem sog. Gast- oder Wirtsrechner entwickelt wird.

2.2 PEARL

Für MRCA/Tornado wurde in einer Vorstudie untersucht, welche Anforderungen ein Avionik-System an eine höhere Programmierspra-

che stellt. Diesen Forderungen entsprach am ehesten die damals rudimentär vorhandene Sprache PEARL. In einem ersten Schritt wurde daher eine konsistente Beschreibung von PEARL für Avionik-Anwendungen, das Avionic PEARL, erstellt. 1976 wurde eine erste Beschreibung, 1977 eine Revision davon veröffentlicht. Leider kam die Entwicklung eines entsprechenden Kompiliersystems für MRCA/Tornado zu spät. 1979 wurde das Kompiliersystem für Avionic PEARL von Fa. ESG fertiggestellt. Auf Verwendungsfähigkeit wurde es in Zusammenhang mit der Digitalisierung des Kampfpanzers Leopard getestet. Dabei wurde festgestellt, daß PEARL für solche Realzeitaufgaben gut geeignet ist und die üblichen Vorteile höherer Programmiersprachen auch bei solchen Anwendungen voll zum Tragen kommen.

Leider hat der "Avionic Subset" die Entwicklung zu Basic PEARL nicht mitgemacht und stellt - wie PAS2 - einen Exoten in der PEARL-Welt dar.

Um einen genauen Überblick über die existierenden PEARL-Systeme zu erhalten und um festzustellen, welchen generellen Sprachumfang sie erkennen und welche Leistungsfähigkeit sie haben, wurde die Hochschule der Bundeswehr in München beauftragt, einen entsprechenden Vergleich anzustellen. Untersucht wurden die Systeme von Fa. AEG, Fa. GPP, Entwicklungsbüro Werum, Fa. Dornier System, Fa. ESG, Fa. BBC, Fa. DEC und Fa. Siemens.

Das Resultat war:

- Reifegrad

  Alle Kompiliersysteme weisen einen mittleren bis guten Reifegrad auf.

- Handhabung

  Die Handhabung der Systeme ist zum Teil sehr schwierig und umständlich.

- Effektivität bzgl. Laufzeit und Speicherbedarf

  Die Systeme sind hier sehr unterschiedlich. Manche Sprachkonstrukte werden in einem System sehr gut umsetzt, andere sehr schlecht.

- Sprachumfang

  Alle Systeme haben einen größeren Sprachumfang als Basic PEARL implementiert. Leider ist der größte gemeinsame Nenner jedoch Basic PEARL, da jedes System andere Erweiterungen gegenüber der Norm hat. Portabilität zwischen den Systemen wird sehr eingeschränkt, da die als implementationsabhängig bezeichneten Teile der Norm sehr unterschiedlich implementiert sind.

3. Standardisierung

Im Falle einer Standardisierung von Programmiersprachen in der Bundeswehr müssen verschiedene Randbedingungen beachtet werden, auch wenn außer Zweifel steht, daß bedeutende Vorteile durch einen solchen Schritt erzielt werden.

Solche Randbedingungen sind z.B.

- In der Bw zu lösende Software-Problematiken sind in allen vier eingangs erwähnten Ebenen angesiedelt.

- Es gibt große eingeführte Software-Systeme, die ständig weiter entwickelt werden.

- Aus Kostengründen müssen auch kommerziell erhältliche Software-Produkte eingesetzt werden.

- In Teilbereichen der Bw sind bereits Programmiersprachen als Standard eingeführt.

Bei einem Kurzvergleich zwischen Ada, PEARL und anderen gängigen Realzeit-Programmiersprachen wurde nun folgendes herausgefunden:

- PEARL ist die modernste und am besten geeignete Programmiersprache für Realzeitaufgaben, die zur Zeit verfügbar ist.

- PEARL ist geeignet für die Formulierung realzeit-orientierter Anwenderprogramme, jedoch nicht zur Formulierung der darunterliegenden Ebenen.

- Ada ist am besten geeignet für die Formulierung von Problemen aus dem Bereich der Systemimplementierung und ähnlichem, jedoch nicht für realzeit-orientierte Anwenderprogramme.

Der Präsident des Bundesamtes für Wehrtechnik und Beschaffung entschied daher im Februar 1982, um langfristig die Software-Kosten in Grenzen zu halten, daß bei neuen Vorhaben, die Programmiersprachen

Ada und

PEARL

vorrangig einzusetzen sind. Um bestehende Gegebenheiten zu berücksichtigen, sind folgende Spezialsprachen zugelassen:

- JOVIAL im Luftverteidigungsbereich
- CMS-2 bei Schiffswaffensystemen und zugehörigen landgestützten Anteilen
- ATLAS bei rechnergesteuerten Prüfsystemen

Assembler-Sprachen dürfen grundsätzlich nicht mehr verwendet werden.

Bei alten Vorhaben, bei denen mit der Programmierung noch nicht begonnen wurde, sind anders lautende Entscheidungen zu überprüfen.

Da die Standardsprachen nicht in jedem Fall den Gegebenheiten gerecht werden können, sind eine Reihe von Ausnahmen zugelassen, die jedoch einer besonderen Genehmigung bedürfen. Solche Ausnahmen sind beispielhaft:

- Verwendung komerziell erhältlicher Software-Produkte
- übernationale Interessenskonflikte
- Nichtverfügbarkeit geeigneter Programmiersysteme

4. Ausblick

Bei verschiedenen Projekten hat sich gezeigt, daß Basic PEARL als Sprachumfang zu klein ist. Es wäre wünschenswert, wenn Basic PEARL um folgende Sprachkonstrukte erweitert würde:

- beliebige Reihenfolge von Deklarationen
- Prozedur-Deklarationen auf allen Ebenen
- Initialisierung von Feldern und Strukturen
- verbesserte String-Verarbeitung
- dynamische Feldvereinbarungen
- verbesserte Ansprechbarkeit von Prozeß-Peripherie (z.B. jede "Klemme" einzeln)
- Referenz-Objekte
- strikte Festlegung von bis jetzt als implementationsabhängig bezeichneten Punkten der gültigen Norm

Diese Erweiterungen von Basic PEARL sollen nicht mit der Definition eines "Defense PEARL" erzwungen werden, sondern sollen durch enge Zusammenarbeit mit anderen Nutzern von PEARL als Vorschläge in die mit der Normung beauftragten Gremien eingebracht werden. Ein Forum zur Erarbeitung solcher Änderungsvorschläge ist der Anwender-Ausschuß im PEARL-Verein, in dem das BWB seit Beginn an vertreten ist.

Fickenscher, Gustav
Bundesamt für Wehrtechnik und Beschaffung
Konrad-Adenauer-Ufer 2 - 6
5400 Koblenz

SASA – Schießausbildungs-, Sicherheits- und Auswerte-Anlage

von Dr.-Ing. **Horst Weber,** Frankfurt am Main

Zusammenfassung

Die Schießausbildungs-, Sicherheits- und Auswerteanlage ist ein größeres Automatisierungssystem, basierend auf mehreren gekoppelten Prozeßrechnern.
Zur Entwicklung der Anwendersoftware wird im Battelle-Institut e. V., Frankfurt, Basis-PEARL als Programmiersprache eingesetzt.
Es werden die Anforderungen an die Software und an ihre Entwicklung beschrieben. Die Probleme beim ausschließlichen Einsatz von PEARL und ihre Lösung durch besondere Programmiermethoden und kleinere Assemblerprogramme werden an Beispielen erläutert.

Schlüsselwörter: gekoppeltes Mehrrechnersystem, Schießplatzautomatisierung, universelles Dialogsystem in PEARL

Summary

SASA, a computerized training and security control for an air defense artillery range, is an extensive automation system based on a hierarchical realtime computer network. Battelle-Institut e. V., Frankfurt am Main, is developing the application software in PEARL.
The requirements to the software and to the software development are described. Problems which arise by exclusive usage of PEARL and its solutions by means of special programming methods and small assembler programs are described and examples are given.

Keywords: computer network, automation system for artillery range, universal dialog system with PEARL

1 Übersicht über Anforderungen, Hardware- und Software-Komponenten der Anlage

1.1 Anforderungen

Die SASA-Anlage soll den heute größtenteils manuell gesteuerten und überwachten Schießbetrieb auf dem Flugabwehrschießplatz der Bundeswehr in Todendorf an der Ostseeküste automatisieren.

Dies ist notwendig - zum einen, weil die Anforderungen an verfügbare Schießzeit, Variierbarkeit der Übungen und Geschwindigkeit der Flugziele ständig erhöht werden, und zum anderen, weil durch die Eigenschaften moderner Waffensysteme, wie große Reichweite, Richt- und Feuergeschwindigkeit, die Sicherheitsüberwachung mit dem herkömmlichen Methoden nicht mehr gewährleistet werden kann. Weiterhin möchte man die Auswertung und Anzeige der Schießergebnisse verbessern und beschleunigen, damit die leitenden Offiziere unmittelbar die Leistungen der Soldaten beurteilen können.

Die Geschütze werden zum Schießen in parallel zur Küste angelegten Schießbahnen in Stellung gebracht und feuern in Richtung See. Da die Reichweite meist über die Dreimeilenzone hinausgeht, kann jeweils nur in solche Richtungen geschossen werden, für die eine Gefährdung von eventuell vorbeifahrenden Schiffen unter Verwendung genügend großer Sicherheitssektoren ausgeschlossen werden kann (B i l d 1).

Zur Zeit erfolgt die Überwachung der Richtung visuell durch einen Soldaten, der gegebenenfalls das Feuer manuell oder elektrisch unterbrechen kann.

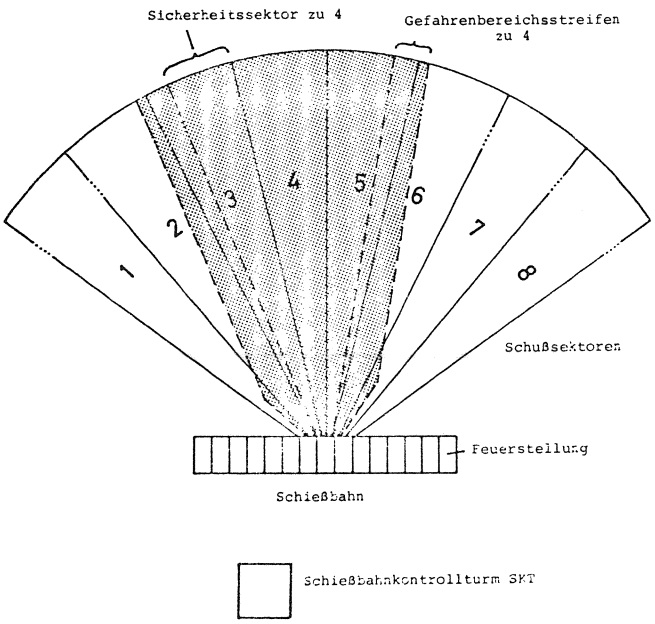

Bild 1. Sektoren einer Schießbahn und Sektorgrenzen für den Sektor 4

Zur Realisierung der Nachrichtenketten während des Schießens sind hintereinander bis zu fünf Soldaten notwendig, um zum Beispiel eine Feuerstop-Nachricht bis hin zu einem Geschütz zu übertragen. In Zukunft wird jedoch weniger Personal zur Verfügung stehen, während die Anforderungen wie oben geschildert zunehmen.

Mit Hilfe der SASA-Anlage soll der Betrieb weitestgehend automatisch ablaufen. Im wesentlichen sind dazu die folgenden Eigenschaften zu realisieren:

- Für den Aspekt Sicherheit wird eine Radarüberwachung des Schießplatzes, des Seegebietes und des Luftraumes durchgeführt.
- Die Ergebnisse der Überwachung werden automatisch schießbahnspezifisch in Informationen über freie und gesperrte Sektoren umgesetzt und den betreffenden Geschützen gemeldet.
- Die Sektorüberwachung bei den Geschützen läuft automatisch ab.

Bei Gefahr für die innere oder äußere Sicherheit kann das Schießen zentral unterbrochen werden.

Die Schleppflugzeuge werden automatisch in ihrem Kurs überwacht und so geführt, daß sich die Schleppziele auf reproduzierbarem Übungskurs an der Feuerstellung vorbeibewegen.

Der Zeitpunkt des Geschoßaustritts und des -durchgangs am Ziel wird automatisch ermittelt und ergänzt durch Angaben über das verursachende Geschütz der Auswerteanlage zur Verfügung gestellt. Auswertedaten sollen unmittelbar nach ihrer Entstehung angezeigt und später in Listen aufbereitet ausgegeben werden.

An zentraler Stelle soll eine Überwachung der gesamten Anlage mit Hilfe visualisierter Daten ermöglicht werden, die dem verantwortlichen Sicherheitsoffizier Steuerung und Beurteilung des Gesamtschießbetriebes erlauben.

Zentrale Anforderung an die SASA-Anlage ist die Gewährleistung der Sicherheit und Zuverlässigkeit in allen Betriebssituationen. Zum Erreichen dieser Eigenschaften wurde vom Auftraggeber als Programmiersprache PEARL verlangt und für die Entwicklung besondere Maßnahmen bei Entwurf und Implementierung der Software vorgeschrieben. Projektbegleitend ist die Dokumentation der entwickelten Software nach vorgegebenen Richtlinien zu erstellen.

1.2 Hardware-Komponenten

Die Hardware-Komponenten der SASA-Anlage sind über den gesamten Schießplatz so verteilt, daß die Prozeßrechner möglichst nahe bei der Peripherie und den Bedienstationen in massiven Gebäuden untergebracht werden können (B i l d 2). So befindet sich der zentrale Auswerterechner R30 in einem Betriebstechnikgebäude nahe dem Hauptkontrollturm (HKT) zusammen mit dem Zentralrechner R10 des Sicherheitsnetzes. Beim HKT werden auch die Radaranlagen mit ihren Rechnern installiert.

Jede der fünf Schießbahnen, die an die SASA-Anlage angeschlossen werden, verfügt über einen Schießbahnkontrollturm (SKT). Darin werden die Sicherheitsrechner SKT untergebracht.

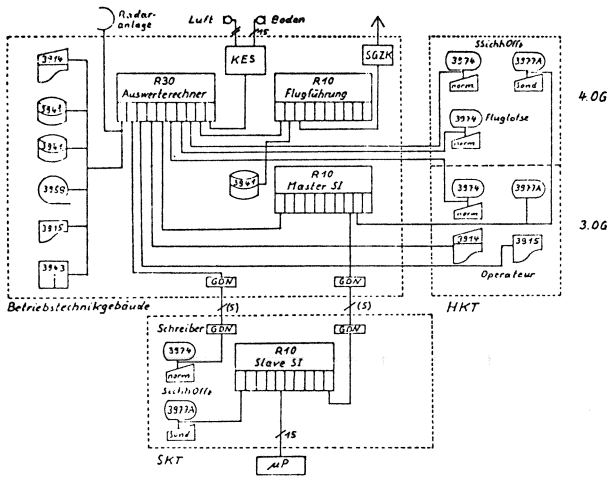

Bild 2. Hardware-Komponenten der SASA-Anlage

Bei den Feuerstellungen in einer Schießbahn werden die Einrichtungen Feuerstellung EF in Geräteboxen untergebracht. Eine EF besteht aus einem Mikroprozessor Intel 8086, einem Mikrophon zur Erfassung des Mündungsknalls, Laser-Winkelmesser, Waffenadapter, Signaleinrichtung und Eingabeeinheit.

Als Ein-Ausgabegeräte dienen Schwarzweiß- und Farbterminals mit Tastaturen, Schnelldrucker und Konsolfernschreiber. Die einzelnen Komponenten sind über Telefonvierdrahtleitungen verbunden, falls nötig mit Nahmodems ausgerüstet.

1.3 Software-Komponenten

In diesem Zusammenhang werden die Radarrechner und die Einrichtungen Feuerstellung nicht beachtet, die für die betrachtete Prozeßrechenanlage nur als Peripherie aufgefaßt werden.

Die Anwendersoftware läßt sich in mehrere Komponenten gliedern, von denen einige je nach Ausbaustufe der Gesamtanlage und je nach Betriebszustand zu- oder abgeschaltet werden können (B i l d 3). Jede Komponente besteht aus mindestens einer Task.

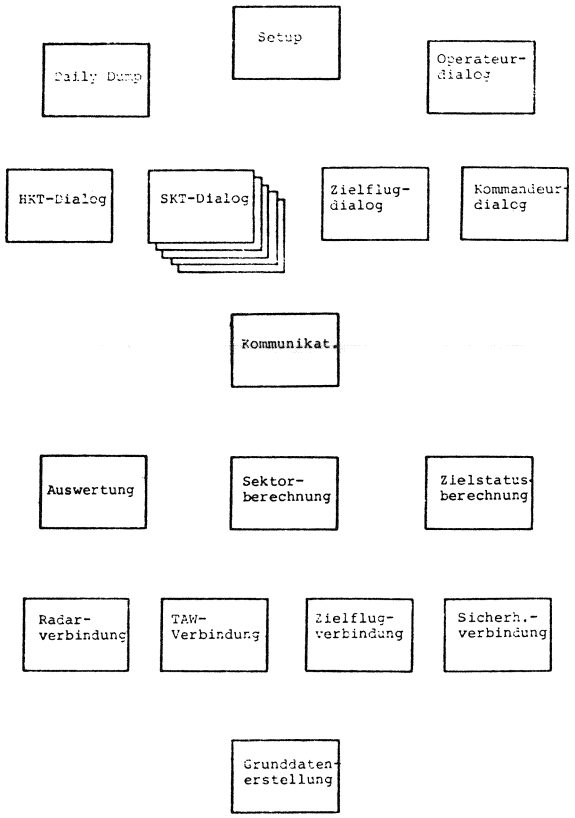

Bild 3. Software-Komponenten der SASA-Anlage

Als zentrales Steuerprogramm für den Gesamtbetrieb dient das Setup-Programm, mit dessen Hilfe auch der Operateurdialog geführt wird. Dies ist ein einfacher Dialog, der hauptsächlich an der Operateurkonsole geführt wird. Er erlaubt das Starten der übrigen Tasks in Abhängigkeit von gewünschten Betriebszuständen, zum Beispiel morgendlicher Neustart des Systems, Wiederstart nach Rekonfiguration oder Netzausfall und das Initialisieren der Listenausdrucke mit den Tagesprotokollen.

Den Arbeitsplätzen des Bedienpersonals sind Dialogprogramme zugeordnet, die universell so entwickelt wurden, daß durch Listensteuerung individuell zugeschnittene Dialoge geführt werden können [3]. Hierzu sind jedem Dialog eventuell mehrere Bildschirmformate zugeordnet (B i l d 4).

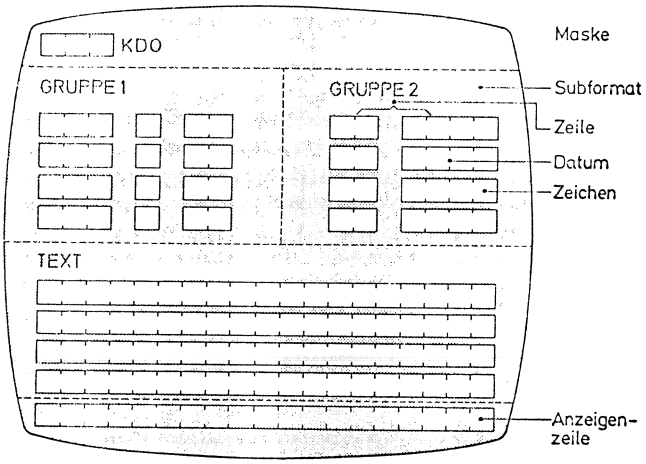

Bild 4. Verallgemeinerte Dialogmaske

Ein Bildschirmformat besteht aus einem Kommandofeld und Subformaten. Ein- und Ausgaben in Subformaten korrespondieren mit speziell zugeordneten Dateien des Datensystems. Subformate sind in logische Eingabeseiten gegliedert und diese wiederum in Einzelzeichen. Nach Eingabe eines Zeichens, bei Abschluß einer Zeile, eines Subformats oder bei Formatwechselkommando kann eine beliebige Nachbearbeitung erfolgen, je nachdem, ob der Eintrag in eine Datei, das Absenden steuernder Nachrichten mit Hilfe des Kommunikationssystems, Formatwechsel, Rollen oder Blättern erfolgen soll.
Das Layout eines Formates, die Zuordnung von Attributen zu den Eingabezeichen und die Abschlußbearbeitung können mit Hilfe spezieller Editoren off line erfolgen.

Das Dialogsystem erlaubt nach seiner Implementierung die vom Nutzer gewünschten Überwachungsfunktionen. So kann sich der verantwortliche Sicherheitsoffizier lesend in andere Dialoge einschalten. Über einen speziellen Dialog können z. B. die Kommandeure einer schießenden Einheit die Schießergebnisse ihrer Soldaten kontrollieren. Derartige Zugriffsrechte sind ebenfalls durch die Art der Nachbearbeitung auf der Basis vorheriger Attributzuordnung möglich.
Die zeichenweise Orientierung der Dialoge erlaubt die Unterbrechung der Eingabe an beliebigen Stellen (B i l d 5). Durch Aufteilung eines Dialogs in Eingabetask und Verarbeitungstask und das Einrichten einer Warteschlange im Empfangsteil der Verarbeitungstask wird es möglich, die Warteschlange sowohl mit Eingabezeichen als auch mit Fehlermessages zu füllen.

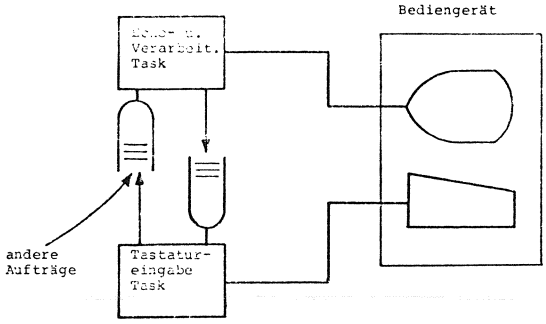

Bild 5. Funktionsprinzip der Dialogeingabe

Die bestehenden Forderungen nach Sicherheit gegen willkürliche oder fahrlässige Fehlbedienungen der Dialoge und nach sofortiger Ausgabe von Fehlermeldungen konnten somit erfüllt werden.

Das Dialogsystem besitzt gegenüber der Datenhaltung eine einheitliche Schnittstelle, die es erlaubt, aus verschiedenen Dations auf Magnetplatten gezielt Datensätze zu lesen und zu schreiben. Durch verschiedene Zugriffsparameter können Dateien unterschiedlich betrieben werden. So werden z. B. Dateien benötigt, die Random-Zugriff auf alle Datensätze erlauben, während andere für lesenden Zugriff nur einen aktuellen Datensatz zurückgeben, für schreibenden Zugriff jedoch einen neuen Datensatz anlegen. Die weiteren Tasks im Auswerterechner bedienen sich der Datenhaltung über die gleichen Schnittstellen. Von besonderer Bedeutung sind dabei die Tasks des Auswertesystems, die aus den von Treibern bereitgestellten Daten aus der Peripherie und über die Datenhaltung zur Verfügung stehenden Grunddaten die Auswertung der Schießergebnisse ermöglichen, wobei mittels der vom Radarrechner gemeldeten Zielkoordinaten eine Rekonstruktion des Schießverlaufs und eine differenzierte Trefferzuordnung beim simultanen Schießen mehrerer Waffen ermöglicht wird. Die Schießergebnisse werden auf den Bildschirmen der entsprechenden Dialoge angezeigt und in Dateien abgelegt. Nach Beendigung der Zuteilung der Schießbahn zu einer Einheit werden bei Beendigung des zugeordneten Schießbahndialogs die Inhalte der Dateien in Listen aufbereitet über Schnelldrucker ausgegeben.

Eine weitere Task berechnet aus den Koordinaten der vom Radargerät erfaßten Objekte auf See und in der Luft die für die einzelnen Schießbahnen zu sperrenden Sektoren. Eine Reihe von Input- und Outputtreibern bedient die jeweiligen seriellen oder parallelen Schnittstellen zur Peripherie. Die ankommenden oder abgehenden Daten sind in Puffern enthalten, wobei die Zugriffe über Semaphore und zugeordnete Kennfelder geordnet werden.

2 Erfahrungen bei der Programmierung mit PEARL

2.1 Vorgehensweise bei der Entwicklung

Die Entwicklung der Software basiert auf einer zusammen mit dem zukünftigen Nutzer entworfenen Funktionsspezifikation, in der die Abläufe, Bildschirmlayouts, Listen und Datenkataloge festgelegt wurden. Aus den Datenkatalogen wurden Jackson-Datenstrukturen entwickelt. Mit ihrer Hilfe und mit Funktionsdiagrammen wurden die Programmspezifikationen top-down entwickelt.

Zur Dokumentation der Programmspezifikationen wurde ein Pseudocode verwendet, der im Hinblick auf die PEARL-Programmierung die strukturierenden Ausdrücke dieser Sprache enthält. Diese Dokumentation wurde mit Hilfe von Editoren auf dem Entwicklungsrechner R30 durchgeführt.
Auf dem Niveau der Pseudocode-Programme wurden vom Projektteam die sicherheitsrelevanten Abläufe in Walkthroughs kontrolliert.

2.2 Programmierung der PEARL-Programme

Mit Hilfe des Editors konnten aus dem Pseudocode die Quellenprogramme entwickelt werden. Der verwendete Sprachumfang ist Basis-PEARL mit den Erweiterungen, die PEARL-300 von der Firma Siemens [1] zur Verfügung stellt. Bezüglich der Sprache gab es für das Programmierteam dabei keine besonderen Schwierigkeiten.

2.3 Maßnahmen für Sicherheit und Zuverlässigkeit

Zu den Bedienern der Anlage gehört an den Bildschirmterminals teils wenig geschultes, teils länger dienendes, besonders eingeübtes Personal. Zur Gewährleistung der Sicherheit dienen die besonderen Schutzmaßnahmen in den Dialogen mit enger Bedienerführung. Hierzu sind unter anderem auch einige Tasten der serienmäßigen Tastatur zu blockieren.

Zum Erlangen größtmöglicher Fehlerfreiheit dienen während der Entwicklung verschiedene Kontrollen, die innerhalb des Projektteams gegenseitig durchgeführt werden. Zur dynamischen Kontrolle sind jedoch die den einzelnen Softwaremodulen zugeordneten Testrahmen unerläßlich, die während der Entwicklung notwendig sind, aber auch später für Tests auf allen Integrationsstufen der Anlage zur Verfügung stehen. Die Testrahmen beinhalten einfache Ein- und Ausgabetasks zur Bereitstellung und Kontrolle von Daten und weiterhin Simulationstasks zur Kontrolle von zeitlichen Verläufen und Einstellung verschiedener Belastungsstufen. Zur Programmierung der Testrahmen eignete sich PEARL in hervorragendem Maße, insbesondere wegen der verschiedenen Möglichkeiten, über Zeitbedingungen den Lauf der Tasks zu beeinflussen.

2.4 Besondere Probleme und jeweilige Lösungen

Die in diesem Zusammenhang aufgeführten Probleme sind primär keine Probleme der Programmiersprache PEARL. Nach unseren Erfahrungen ist es allerdings nicht möglich, nur mit PEARL-Kenntnissen ein solch komplexes System wie die SASA-Anlage zum Laufen zu bringen. Im folgenden sind einige Beispiele für die aufgetretenen Probleme aufgeführt.

Über das Laufzeitsystem werden in dem verwendeten PEARL-System einige mathematische Funktionen, Zeit- und Datumsabfragen usw. zur Verfügung gestellt. Zeit- und Datums-

eingabe fehlen dabei. Als besonders wichtig erweisen sich in diesem Fall die genaue Dokumentation des Parameterübergabemechanismus und die Manipulation des Stacks. Dann ist es mit Hilfe kleiner Assembler-Routinen möglich, die Library zu vervollständigen.

Der für die Dialoge konzipierte zeichenweise Betrieb einer Eingabetask ist in PEARL nicht vorgesehen, d. h. es stehen keine entsprechenden Standardtreiber zur Verfügung. Mit Hilfe einer kleinen Assembler-Routine kann der gewünschte Betrieb zwar realisiert werden, die zugehörige Task muß aber nunmehr hauptspeicherresident geladen werden. Nun stören die nicht wiedereintrittsfähigen Laufzeitroutinen sehr, die zu jeder Task hinzugebunden werden müssen. Andere Treiber benötigen für ihr Funktionieren spezielle Speicherbereiche (sogenannte reelle Pakete), die wegen anderer Zwänge nur geringen Umfang besitzen. Die PEARL-Tasks lassen sich nur unter Beachtung vieler Randbedingungen funktionsfähig dorthin laden.

Sobald PEARL-Tasks gemeinsame Daten im Hauptspeicher benutzen, muß ein gemeinsamer Globalbereich existieren. Andererseits können Tasks mit ihren Prozeduren bei gemeinsamem Zugriff auf Daten entweder die Parameterübergabe benutzen oder globale Daten verwenden. Falls sehr viele Parameter zu übergeben wären, wird hierfür viel Code und Laufzeit nötig. Falls die Daten global gemacht werden, müßten sie allerdings dem allgemeinen Globalbereich zugefügt werden, der im Falle der SASA-Anlage sehr groß geworden wäre, was Nachteile für die Speicherverteilung, Testbarkeit und Handhabbarkeit des Systems gehabt hätte. Zur Vermeidung dieser Nachteile mußten sehr große lineare PEARL-Programme geschrieben werden, deren Handhabbarkeit allerdings schlecht ist.

Der verwendete Compiler berechnet zur Compilezeit keine Konstantenausdrücke. Falls code- oder laufzeitoptimale Programme benötigt werden, ist dies beim Codieren zu berücksichtigen.

Der verwendete Compiler übersetzt Quellenzeilen ohne Berücksichtigung des Kontextes. Aus diesem Grunde sind möglichst viele Operationen vor einer Zuweisung durchzuführen. Dies führt zwangsläufig zu wenig übersichtlichem Quellentext.

3 Ergebnisse bei der Integration der Komponenten

3.1 Testergebnisse

Durch die Vorgehensweise bei der Programmentwicklung ergeben sich relativ wenige Probleme bei der Integration der Komponenten. Durch sukzessiven Ersatz von Testrahmen durch endgültige Komponenten wird das System vervollständigt. Die durchgeführten Tests dienen zur Verifikation der Funktionen, wie sie in der Funktionsspezifikation festgelegt wurden. In diesem Rahmen konnte ein sehr großer Teil des Tests, die ohne Radargerät und Trefferauswerteanlage durchführbar sind, zufriedenstellend durchgeführt werden.

3.2 Zeitverhalten

Das Zeitverhalten der EDV-Komponenten ist von der Aufteilung des Hauptspeichers in die "virtuellen" Laufbereiche abhängig. Programme in reellen und anderen Paketen und die globalen Prozeduren stehen ständig zur Verfügung. Programme in den Laufbereichen unterliegen einer Überwachung und werden nach Bedarf in Abhängigkeit von Prioritäten bei E/A-Anforderungen und Einplanungen ein- oder austransferiert. Die Verteilung der Tasks des Systems auf Laufbereiche und Pakete beeinflußt in starkem Maße das Zeitverhalten. Zur Optimierung werden umfangreiche Tests mit dem fertigen System notwendig sein.

3.3 Fehlerbeseitigung

Die Fehlersuche und -beseitigung in den separat betriebenen Modulen ist mit Hilfe von zur Verfügung stehenden Dienstprogrammen relativ einfach möglich. Die

Dienstprogramme arbeiten jedoch auf Assembler- bzw. Grundspracheniveau, so daß ein PEARL-Programmierer darüber Kenntnisse haben muß.

Im Falle, daß Programmsysteme getestet werden müssen, erweist es sich als unpraktisch, daß Semaphore von einem PEARL-Programm nicht vorbesetzt werden können. Da sie im Commonteil deklariert sind, können sie nur durch Neuladen auf definierte Anfangswerte gesetzt werden - für Tests beim Programmsystem eine oft unerwünschte Vorgehensweise.

Im Projekt erweist sich eine spezielle Hilfstask als besonders notwendig, die mit Hilfe von Assembler-Programmen Manipulationen in den common geladenen Datenbereichen ermöglicht. Sie spart erheblichen Zeitaufwand in der Test- und Integrationsphase ein.

4 Ausblick, Erwartungen

Im Gegensatz zu den Zielen der PEARL-Sprachentwicklung läßt sich ein Programmsystem zur Zeit nicht ohne Betriebssystem-, Assembler- und Hardware-Kenntnisse des PEARL-Programmierers zum Ablauf bringen. Falls ständig mindestens ein Mitglied des Projektteams über diese Kenntnisse verfügt, läßt sich ein Programmier- und Testbetrieb nach "Kochrezepten" aufziehen, wobei die Beschränkung auf PEARL-Sprachkenntnisse bis zu gewissen Grenzen möglich ist.
Unbedingt notwendig wäre für diese Arbeitsweise eine Testhilfe auf Sprachebene.
Zur Vermeidung der Assembler-Ergänzungsprogramme wäre die Existenz von Libraries notwendig, die über das heutige Maß hinaus die notwendigen Prozeduren enthalten sollten. Sie sollten mit dem Compiler gekauft werden können und mit entsprechender Dokumentation versehen sein.

Schrifttum

/1/ Siemens: Beschreibung PC30/PEARL 300. Siemens P71100-D3010-X-A3-35

/2/ Weber, Horst: Einsatz von PEARL bei der Software-Entwicklung für eine Fla-Schießplatz-Automatisierung. PEARL-Rundschau (1980) Nr. 4, S. 26/31

/3/ Weber, Horst und Egner, Karl-Dieter: Ein modulares, universelles Dialogsystem programmiert in PEARL. PEARL-Rundschau (1981) Nr. 6. S. 31/39

Anschrift des Autors

Dr.-Ing. Horst Weber
Battelle-Institut e. V.
Abt. Technische Informatik
Am Römerhof 35

6000 Frankfurt am Main 90

Tel. (0611) 7908-2508

ARES – Artillerie Raketen Einsatzsystem

von **D. Krönig,** Friedrichshafen

Zusammenfassung

ARES ist das Feuerleitsystem der Raketenartillerie auf Batterieebene. In Vorbereitung auf die Entwicklung der ARES-Software sind Untersuchungen zur Datenhaltung bei der Programmierung in PEARL angestellt worden. Es zeigt sich, daß die ARES-Anforderungen am besten durch ein PEARL-Datenbanksystem abgedeckt werden. In einer prototypartigen Erprobung mit ARES-Abläufen erweist sich das PEARL-Datenbanksystem BAPAS-DB als geeignet. Abschätzungen des statischen und dynamischen Verhaltens liefern befriedigende Ergebnisse.

Summary

ARES is the fire control system of the rocket artillery at battery level. During preparation for the development of the ARES-Software investigations concerning the file-management when programming in PEARL have been conducted. It is indicated that the requirements of ARES would be best met by a PEARL-DBMS. In a prototype implementation some ARES-functions were examined. The PEARL-DBMS BAPAS-DB turned out to be qualified. Estimates of the static and dynamic behaviour produce satisfactory results.

1. Einleitung

ARES ist das Feuerleitsystem der Raketenartillerie auf Batterieebene. Die Definitionsphase von ARES wurde 1981 abgeschlossen. In Vorbereitung auf die Entwicklungsphase sind einige weiterführende Untersuchungen zur Soft ware von ARES angestellt worden. Besonderes Augenmerk wurde dabei auf die Datenhaltung bei der Programmierung in PEARL gerichtet. Grundsätzlich bietet PEARL mit der Dateiverwaltung des unterliegenden Betriebssystems die Möglichkeit, Datenhaltungsaufgaben ohne weitere Hilfsmittel zu lösen. Allerdings sind dann Funktionen wie Konsistenzsicherung, konkurrierende Zugriffe, Sicherung gegen Verlust etc., die ein Datenbanksystem zur Verfügung stellen kann, explizit zu programmieren. Es ist also im Einzelfall zu prüfen, ob ein Datenbanksystem existiert, das den Projekterfordernissen genügt. Hierzu gehören neben den funktionalen Anforderungen die Rahmenbedingungen (Rechner, Programmiersprache u.a.), einzuhaltende Reaktionszeiten, Kosten, Liefertermine etc...

Die Anforderungen von ARES an die Datenhaltung wurden in diesem Sinne untersucht. Es zeigte sich, daß es ein geeignetes PEARL-Datenbanksystem gibt. In einer Prototyp-Erprobung wurden einige ARES-Funktionen implementiert. Programmläufe und Abschätzungen des statischen und dynamischen Verhaltens lieferten befriedigende Ergebnisse.

2. ARES-Software

ARES hat die Aufgabe, die taktische und technische Feuerleitung der Waffensysteme

- o Mittleres Artillerie Raketen System MARS (MLRS)
- o Leichtes Artillerie Raketen System LARS

wirkungsvoll zu unterstützen, um die Möglichkeiten dieser Waffensysteme - vor allem hinsichtlich der Reaktionszeit - optimal zu nutzen. Zur Erfüllung dieser Aufgabe müssen eine große Anzahl von Befehlen, Meldungen, Übersichten und Orientierungen verzugsarm erfaßt, verarbeitet und weitergeleitet werden.

Die Aufgaben lassen sich zwei Klassen zuordnen:

(1) Empfangen und Senden, Darstellen und Erstellen von Nachrichten (Kommunikationsteile)
(2) Verwalten und Speichern von Daten, Führen der Betriebszustände (Verwaltungsteil).

Dementsprechend ist in der Batteriefeuerleitstelle ein System vorgesehen, das aus einem zentralen Rechner (Batterie-Rechner) für die letztgenannten Aufgaben und aus intelligenten Daten-Ein-/Ausgabe-Geräten (DEA's) für die erstgenannten Aufgaben besteht (s. Bild 1).
(B i l d 1).

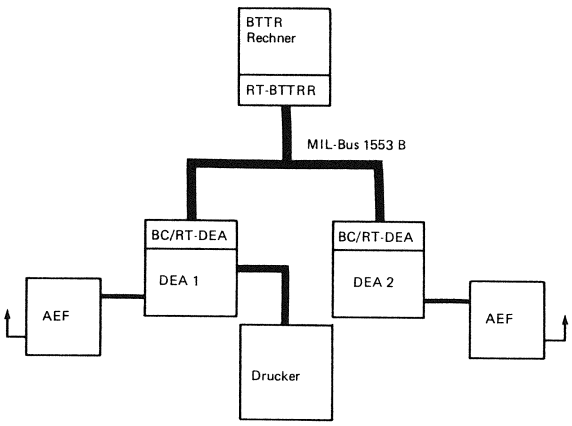

Bild 1. ARES - Batteriefeuerleitstelle

Die DEA's sind über den MIL-BUS an den Batterie-Rechner angeschlossen. Datenfunkverbindung zum Bataillon einerseits und zu den Feuerstellungsräumen andererseits besteht über die DEA's (Anschluß an Datenfunk)

Die wichtigsten Funktionen und Abläufe im zentralen Rechner sind in Bild 2 dargestellt (B i l d 2).

Über den MIL-BUS (linke Seite) gelangen die Nachrichten vom DEA in Form von Datentelegrammen in den Rechner. Sie werden dort von der Telegrammverwaltung angenommen und über die Auftragsverwaltung den jeweiligen Aufgaben zugeordnet.

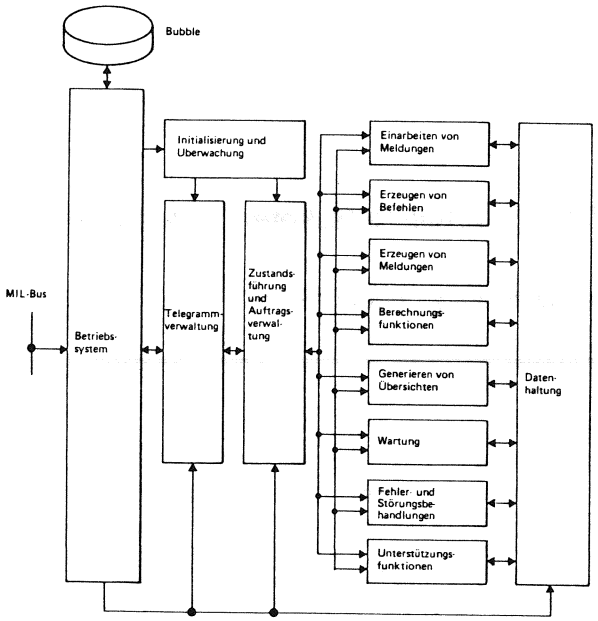

Bild 2. ARES - Software-Struktur Batterie-Rechner

3. Datenverwaltung

Eine zentrale Stellung nimmt, wie Bild 2 zeigt, die Datenverwaltung ein. ARES stellt daran ersichtlich folgende besonderen, funktionalen Forderungen:

- o Eignung für Echtzeitbetrieb, resultierend aus nebenläufigen Vorgängen und schritthaltender Verarbeitung
- o Verwaltung konkurrierender Zugriffe auf die Daten, wie Bild 2 zeigt
- o Gewährleistung der Konsistenz der gespeicherten Daten
- o Sicherung der Daten gegen Verlust
- o Einfache Handhabung durch den Bediener

Hinzu kommen die Randbedingungen der

- o PEARL-Konformität und
- o AEG 80-20/6 M-Verfügbarkeit.

Zur Realisierung bieten sich grundsätzlich zwei Lösungen an:

- o PEARL-Dateiverwaltung
- o PEARL-Datenbanksystem.

Die Verwendung der über PEARL ansprechbaren

Dienste der Dateiverwaltung der unterliegenden Betriebssystems bedeutet, mindestens die oben genannten Funktionen gesondert zu programmieren. Zudem entsteht dabei eine spezielle Lösung, die auf mögliche ähnliche Projekten nur schwer zu übertragen ist.

Demgegenüber liegen die Vorteile eines Datenbanksystems, das obige Anforderungen erfüllt, darin, die Erweiterungs- und Verallgemeinerungsfähigkeit der Software zu fördern, die Transparenz, Nachvollziehbarkeit und Wartbarkeit der Software zu erhöhen und die Dokumentation zu erleichtern. Zudem kann das Entwicklungsrisiko gemindert werden.

Als ein geeignetes Datenbanksystem bot sich das BAPAS-DB der Fa. Werum, Lüneburg, an. Es erwies sich als industriell verfügbar, wie sein Einsatz in einer Anlage zur Produktionsüberwachung einer Fließbandstraße eines Automobilwerkes zeigte. Dabei war BAPAS-DB übrigens von einem Systemhaus, dem IITB der FhG, in das Projekt integriert worden, analog dem Vorgehen bei der möglichen Verwendung durch den GU im Projekt ARES.

BAPAS-DB ist jedoch noch nicht auf AEG-80/6 M verfügbar. Es wurden daher Untersuchungen auf dem PEARL-System der AEG 80-20/6 M angestellt mit dem Ergebnis, daß BAPAS-DB mit begrenzbarem Aufwand auf diesem Rechner implementiert werden kann.

Die anderen oben aufgestellten Forderungen erfüllt BAPAS-DB. PEARL-konform heißt dabei, daß BAPAS-DB aus PEARL als Gastsprache heraus ansprechbar und selbst in PEARL geschrieben ist. Damit ist eine PEARL-einheitliche Programmierung sichergestellt.

Die Forderung nach der einfachen Handhabung durch den Bediener wird dadurch unterstützt, daß Datenbestände in BAPAS-DB konkurrierend aus Programmen heraus und im Dialog angesprochen werden können (s. Bild 3) (B i l d 3).

Ersteres vermittelt die Datenmanipulationssprache DML, letzteres die Abfragesprache QL.

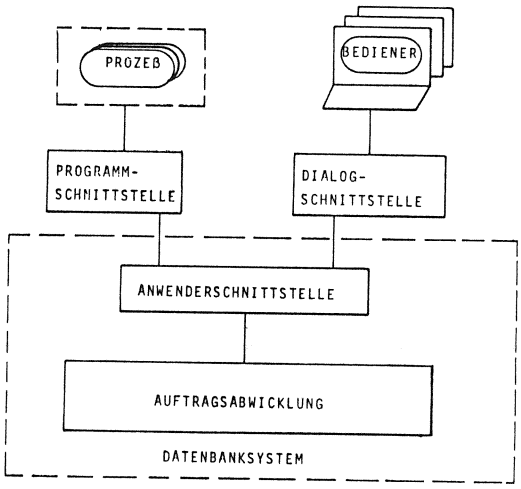

Bild 3. Benutzerschnittstellen zum Datenbanksystem BAPAS-DB

4. Erprobung

Die ARES-spezifischen Probleme wurden an Hand einer Probeimplementierung typischer Abläufe untersucht. Zur Veranschaulichung ist in Bild 4 der Ablauf vom Erkunden des Feuerstellungsraumes bis zum Ende des Feuerauftrages dargestellt.
(B i l d 4).

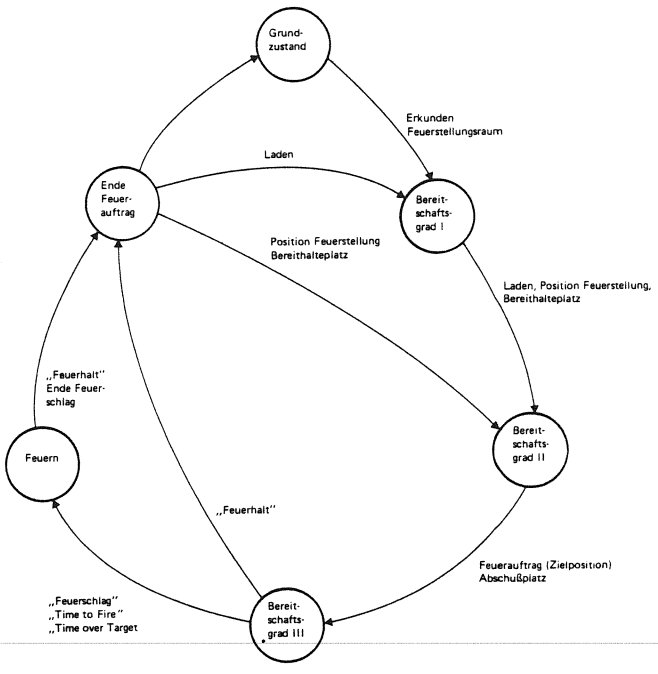

Bild 4. ARES - Funktionsablauf Bereitschaftsgrade

Aus dem unteren Teil, von Bereitschaftsgrad II bis zum Ende Feuerauftrag, wurden hieraus einige Befehle und Meldungen ausgewählt, und zwar:

- o Bearbeitung eines Feuerauftrages
- o Generieren von Call-For-Fire-Befehlen
- o Mission-fired Meldung
- o Bearbeitung eines Feuerhaltbefehls.

In der Funktionsübersicht von Bild 2 sind diese Befehle und Meldungen den Einheiten "Erzeugen von Befehlen" und "Erzeugen von Meldungen" zuzuordnen. Von besonderer Bedeutung ist der Feuerhaltbefehl, weil er vorrangig weiterzuleiten ist. Er kann, wie aus Bild 4 ersichtlich, den Feuerschlag aus Bereitschaftsgrad III heraus abbrechen.

Die Befehlsabläufe und die zugehörigen Dateien waren bereits in der Definitionsphase spezifiziert worden, die Dateien in einer an Pascal angelehnten Notation. Die Übertragung in die PEARL-konforme DDL bereitete wegen des STRUCT-Typs von PEARL keine Schwierigkeiten. Satzdefinitionen werden in BAPAS-DB aus Strukturen aufgebaut. Die Abläufe selbst konnten unmittelbar in PEARL übertragen werden. Die Dateien können per Programm oder direkt und konkurrierend über die Query Language QL im Dialog angesprochen werden (s. Bild 3). Das Programm ist ablauffähig auf dem Rechner Nord-10 bei Fa. Werum in Lüneburg sowie auf dem Rechner Siemens 330 R 30 bei Dornier in Friedrichshafen.

Es wurde festgestellt, daß die Verwendung eines PEARL-Datenbanksystem die Entwicklung unter den genannten Anforderungen nachhaltig unterstützt und sicherer macht als die Verwendung einer einfachen Dateiverwaltung. Reaktionszeiten, die nicht tolerierbar sind, wurden bei den Versuchen nicht beobachtet. Speicherplatzabschätzungen ergaben, daß im operationellen Betrieb (zur Laufzeit des Zielsystemes) ca. 20 kBytes (einschließlich der Zugriffsverfahren Hash und Fifo) resident benötigt werden, mit Query Language (falls diese zur Laufzeit benötigt wird) ca. 20 kBytes zusätzlich resident.

Da für die Erprobung, um den Aufwand in Grenzen zu halten, wenige ARES-Abläufe herangezogen wurden, ist eine abschließende Beurteilung heute noch nicht möglich. Eine Feststellung kann aber schon heute getroffen werden, daß nämlich die PEARL-konforme Definition und Ansprache einer BAPAS-DB-Datenbank die Programmierung begünstigt.

5. PEARL

Der Nutzen, der aus der Verwendung einer höheren Programmiersprache erwächst, braucht heutzutage nicht mehr betont zu werden. Doch ist nicht jede Sprache für jedes Projekt gleich geeignet. ARES erfordert die Programmierung nebenläufiger Vorgänge ebenso wie numerische Berechnungen, die Kommunikation zwischen Rechenprozessen ebenso wie die Handhabung unterschiedlicher EA-Vorgänge. Hierfür gibt es in PEARL wirkungsvolle Sprachkonstrukte.

Die Entwicklung großer Projekte wie ARES erfolgt arbeitsteilig. Es ist erforderlich, an einer Stelle Geräte herzustellen, die gleichzeitig an anderer Stelle bereits programmiert werden müssen. Hier bietet PEARL durch die Systemteil-/Problemteil-Trennung die Möglichkeit zu realistischen Verabredungen zwischen den Arbeitsteams. Nachdem nämlich der Systemteil des Subsystems gemeinsam festgelegt wurde, kann einerseits programmiert und andererseits das Gerät fertig entwickelt werden. Von solchen Verabredungen wird bei der ARES-Entwicklung Gebrauch gemacht.

Bleiben noch die Anforderungen an die Datenverwaltung unter Echtzeitbedingungen, bei konkurrierenden Zugriffen von Rechenprozessen und vom Bediener, mit Zugriffsschutz- und Konsistenzsicherungsforderungen. Wie gezeigt wurde, gibt es auch hierfür eine Lösung, die voll in die PEARL-Umgebung eingebettet ist.

Pearl erweist sich also als eine dem Projekt ARES angemessene Programmiersprache.

6. Schlußbemerkung

Die Erprobung von BAPAS-DB an einigen Abläufen von ARES wurde gemeinsam von Fa. Werum, Lüneburg, und Fa. Dornier, Friedrichshafen, betrieben. Bei Fa. Werum war H. Blumental damit betraut. Bei Dornier hat H. K. Goerschel die Erprobung durchgeführt.

Anschrift des Autors:

Dr. Dirk Krönig
Dornier GmbH
Postfach 1420
7990 Friedrichshafen

PEARL-System für ein athmosphärisches Meß- und Auswertesystem ATMAS

von Dr. **Fenzel,** ESG München

1. EINLEITUNG

Das Atmosphärische Meß- und Auswerte-System (ATMAS) erstellt formalisierte Wettermeldungen aus meterologischen Meßdaten der freien Atmosphäre. Die Meßgrößen Lufttemperatur, relative Luftfeuchte und Luftdruck werden von einer Radiosonde an einem Freiballon per Funk an ein Aufnahme- und Registriergerät für Wettersonden (ARWET) gesendet. Das Ballonverfolgungsradar vermißt die Ballonbahn und überträgt Meßwerte für Azimut, Elevation und Schrägentfernung des Ballonortes an den Rechner. ATMAS erstellt automatisch die verschiedenen Wettermeldungen: METB2, METB3, METTA, TEMP, PILOT und SW.

2. ATMAS-FUNKTIONEN

Die Benutzung von ATMAS wird mittels eines umfangreichen Dialogprogrammes gesteuert. Dem Bediener stehen eine Reihe von Kommandos zur Verfügung, mit denen er die gewünschten Aufgaben ausführen kann. Mit den Kommandoeingaben kann der Bediener:

- Versorgungsparameter wie Zeit, Ort, meteorologische Bodenarten, benutzte Meßgeräte, usw. eingeben,
- einen Ballonaufstieg starten, abbrechen oder verwerfen (d.h. die Meßwerte im Speicher löschen),
- Meldungen erstellen, ergänzen, korrigieren, protokollieren und eingeben,
- Radardaten im Dialog eingeben.

Nach dem Kommando "Ballonaufstieg starten" beginnt die automatische Meßwerterfassung, d.h. über je eine V24-Schnittstelle werden vom Radargerät und vom ARWET im Sekundentakt Daten an den Rechner gesendet.

Aus diesen Meßwerten für Temperatur, rel. Feuchte, Druck, Azimut, Elevation und Schrägentfernung werden für die einzelenen Meßgrößen 4s-Mittelwerte gebildet. Die 4s-Mittelwerte werden zur Bestimmung von Ausgleichsgeraden mit der Methode der kleinsten Quadrate benutzt, um die Zeitprofile für Temperatur, rel. Feuchte, Druck, Windgeschwindigkeit und -richtung und Höhe zu berechnen.

Die Zeitprofile bleiben bis zum nächsten Ballonstart gespeichert und dienen als Grundlage zur Berechnung der Werte für die verschiedenen Meldungen.

Um auch einen optischen Eindruck über den Verlauf der Vertikalprofile zu bekommen, hat der Bediener die Möglichkeit, sich die Sondendaten sowie Windgeschwindigkeit und -richtung im Abstand von 20s als Momentanwerte "graphisch" auf einem Drucker ausgeben zu lassen. Diese Darstellung der Meßwerte läuft parallel zum Ballonaufstieg, denn die Meßwerte werden nicht abgespeichert.

Verfügt das benutzte Radargerät über keine Schnittstelle zum Rechner für eine automatische Datenerfassung, so kann der Bediener die abgelesenen Radardaten manuell über das Dialoggerät in den Rechner eingeben.

Mit ATMAS können folgende Wettermeldungen erstellt werden:

METB2	Ballistische Meldungen (Flugabwehr)
METB3	Barbarameldung
METCM	Meldung für den Artillerierechner
METTA	Meldung für Zielerfassung
SW	Schichtmittelwindmeldung
TEMP	Aerologische Meldung
PILOT	Höhenwindmeldung

Die Meldungen werden nach den entsprechenden STANAG-Formaten bzw. nach dem WMO-Code erstellt.

Die Grundmeldung, d.h. die Meldungen METB2, METB3 und METCM kann, wenn sie nicht aus eigenen Messungen vollständig erstellt wurde, mittels fremder Meldungen ergänzt werden.

3. SYSTEMKONFIGURATION

Die Systemkonfiguration ist in Abb. 1 dargestellt.

Sie besteht aus:

- dem Rechner AEG 80-20/5M mit einem Speicheraus-

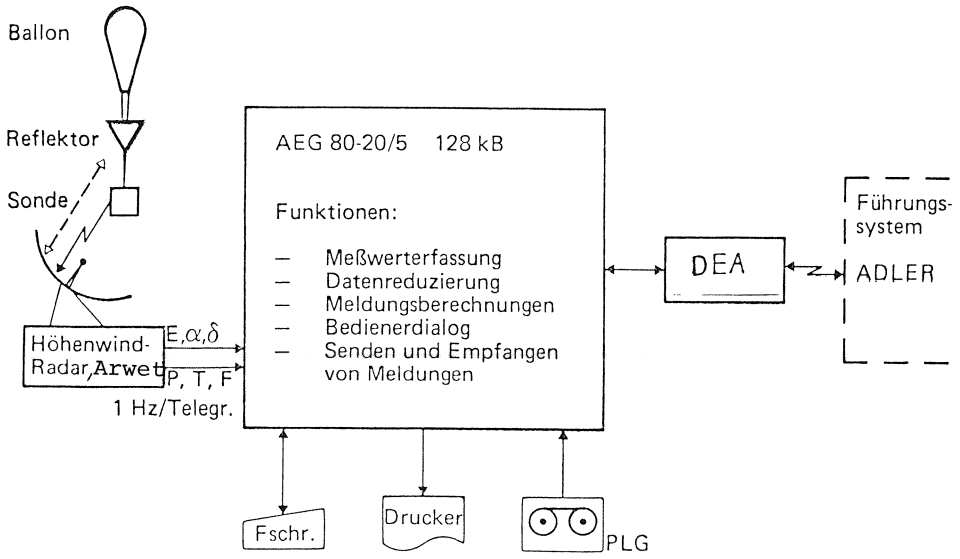

Abb. 1: Systemkonfiguration ATMAS

bau von 128 KB (für späteren Einsatz ist der Rechner AEG 80-20/6M mit 256 KB vorgesehen),
- einer 4-fach V24-Schnittstelle für Radar, ARWET, DEA (Funkein-/ausgabe) und Drucker,
- Multi-AW (Anpaß-Werk) für den Bedienungs-Fernschreiber und dem Programm-Ladegerät (PLG).

Die Software-Entwicklung erfolgte in der höheren Programmiersprache PEARL, wobei aus Speicheroptimierungsgründen einige Routinen (z.B. E/A-Routinen, Formatierungsroutinen) in Assembler programmiert wurden.
Die Software-Entwicklungsumgebung bestand aus einer zivilen Version des Rechners AEG 80-20/5 mit entsprechender Standard-Peripherie (Platte, Drucker, Terminal) und Standard-Software (Echtzeit-Betriebssystem MARTOS-K, AEG-PEARL-Compiler, Makro-Assembler, Editor, PROGA-Test).

In Abb. 2 sind die PEARL-Verarbeitungsmoduln und der Datenfluß im Zusammenwirken mit den E/A-Geräten dargestellt.
Die Pfeile zeigen den Fluß der Daten von den Eingabe-Geräten ARWET, RADAR, Funkschnittstelle (DEA) und Fernschreiber über die PEARL-Verarbeitungs-Moduln Bediener-Dialog (DIALOG), Eingabeaufbereitung (ARWETT, RADART, EADEA), Auswertung (ARWAUS, RADAUS), Meldungserstellung (MELD) und Druckerausgabe (DIAGRAM) zu den Daten-Puffern (ARWEIN, RADEIN, MET, RAD, MITTELA, MITTELR, ZEITPR, MELDWP) und Ausgabe-Geräten Drucker (SDR), Fernschreiber (FSR) und DEA.

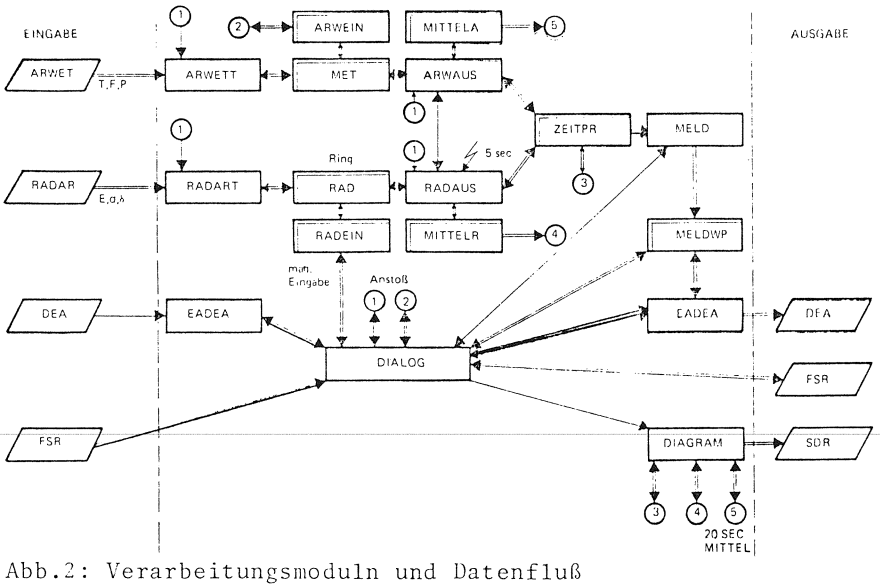

Abb.2: Verarbeitungsmoduln und Datenfluß

Mehrrechner-PEARL für Mehrrechner- und Mehrprozessor-Systeme

von Dr. **H. Steusloff**, Karlsruhe

Zusammenfassung

Mehrrechner- und Mehrprozessorsysteme setzen sich in allen Bereichen der Systemautomatisierung und damit auch im Wehrbereich zunehmend durch. Dabei ist besonders im Wehrbereich eine breite Palette unterschiedlicher Anwendungsumgebungen zu berücksichtigen: Wehrmaterial wird zukünftig ebenso mit Mehrrechner- oder Mehrprozessorsystemen (embedded systems) automatisiert, wie Leit- und Führungssysteme. Die Nutzung der Eigenschaften von PEARL für die Programmierung auch solcher Systeme läßt eine Erweiterung dieser Sprache zur Beschreibung von Strukturen notwendig erscheinen. Zwei erprobte Ansätze für MEHRRECHNER-PEARL (Dornier, Friedrichshafen und IITB, Karlsruhe) werden vorgestellt und diskutiert.

Schlüsselworter: Mehrrechnersysteme, Programmiersprachen, verteilte Systeme

Summary

Multicomputer and multiprocessor systems show increasing importance in all areas of system automation, military systems included. Especially in the defence area a large variety of application environments has to be considered: embedded systems will be automated by multicomputer systems as well as guidance and control systems. To utilize the properties of PEARL in programming these systems, an extension of this language is necessary towards the description of structures. Two approved developments of a MULTICOMPUTER-PEARL (Dornier, Friedrichshafen and IITB, Karlsruhe) are presented and discussed.

Keywords: Multicomputer system, programming languages, distributed systems

1. PROBLEMEINFÜHRUNG

Bedingt durch Forderungen nach erhöhter Signalverarbeitungsleistung sowie äußerster Zuverlässigkeit und Verfügbarkeit von rechnergestützten Systemen im Verteidigungsbereich, werden dort Mehrrechnersysteme in Zukunft zunehmend eingesetzt werden müssen. Solche Mehrrechnersysteme sind durch höhere Programmiersprachen zu programmieren, um die heute vorliegenden Anforderungen bezüglich Wirtschaftlichkeit und Zuverlässigkeit der Programmsysteme zu erfüllen. Die derzeit hierfür besonders geeignete Echtzeitprogrammiersprache ist PEARL (Process and Experiment Automation Realtime Language [1]). Das Modulkonzept und der Systembeschreibungsteil dieser Sprache erleichtert ihren Einsatz für Mehrrechnersysteme mehr als andere derzeit verfügbare Sprachen.

Es fehlen andererseits auch bei PEARL nach DIN 66253 gewisse Sprachmittel, die den Entwurf und die Programmierung von Mehrrechnersystemen wirkungsvoll unterstützen, wie solche zur Systemstrukturbeschreibung oder zur Beschreibung und Steuerung von Konfiguration und Rekonfiguration der Anwendungsprogramme, auch in Abhängigkeit vom Betriebszustand des Rechnersystems. Ebenso ist die Prozeßkommunikation und die Synchronisation von Programmen in verschiedenen Prozessoren zu beachten.

Bei Ergänzungen von PEARL nach DIN 66253 zu Mehrrechner-PEARL ist zu berücksichtigen, daß die Hardware von Mehrrechnersystemen nach

- der Art der Prozessoren
- dem Kommunikationsprinzip und seiner Realisierung sowie nach
- den Methoden zur Erhöhung der Fehlertoleranz

zu unterscheiden ist. Für die Nutzung dieser je nach Einsatzfall unterschiedlich ausgeprägten Merkmale von Mehrrechnersystemen müssen Sprachmittel zur Verfügestellt werden, die dem Anwendungsprogrammierer ein Verbleiben in der höheren Sprache erlauben.

2. EINSATZUMGEBUNGEN FÜR MEHRRECHNER-PEARL

Im militärischen Bereich besteht die Notwendigkeit, Mehrrechnersysteme als Automatisierungskomponenten innerhalb von Gesamtsystemen, z.B. für Avionik und Waffensysteme einzusetzen (embedded systems). Hier-

bei ist wesentlich eine

- hohe Speicher- und Laufzeiteffizienz der Programme,
- Verringerung der Fehlermöglichkeiten bei der Programmierung und
- gute Dokumentation sowie wirtschaftliche Erstellung und Wartung der Programme.

Um komplizierte Kommunikationsverfahren zu vermeiden, wird oft eine möglichst lose Kopplung der einzelnen Prozessoren untereinander angestrebt.

Die Struktur des Gesamtsystems und damit auch seines Automatisierungssystems wird während des Betriebes nicht geändert, d.h. die Flexibilitätsanforderungen sind diesbezüglich gering. Notwendige Fehlertoleranzenmaßnahmen liegen damit für die Lebensdauer des Systems fest und können sowohl durch statische Redundanz der Hardware (m aus n-Systeme) erreicht werden, als auch vom Entwickler je nach den Systemanforderungen in anderer Weise fixiert werden.

Eine andere Einsatzumgebung von MEHRRECHNER-PEARL ist gegeben bei der Verwendung von Mehrrechnersystemen zur Automatisierung technischer Anlagen, deren Struktur und Eigenschaften während des Betriebes auf einfache Weise an neue Anforderungen angepaßt werden müssen (open-ended design). Dies können z.B. sowohl militärische Command, Control and Communication Systems als auch industrielle Anlagen sein. Die Systemkonfiguration und die Fehlertoleranzmaßnahmen müssen dabei leicht, ggf. on-line und automatisch anpaßbar sein. Eine vorübergehende Leistungsdegradation wird aus Wirtschaftlichkeitsüberlegungen zugelassen und deshalb dynamische Redundanz vorgesehen. Damit ist die Kommunikation zwischen den Prozessoren komplizierter. Speicherplatzeffizienz hat eine geringere Bedeutung; auf die Laufzeiteffizienz wird dennoch hoher Wert gelegt.

3. MEHRRECHNER-PEARL-SYSTEME

Aus der geschilderten Situation entstanden auf der Grundlage von PEARL nach DIN 66253 zwei MEHRRECHNER-PEARL-Systeme für die Einsatzerfahrungen vorliegen und die im folgenden charakterisiert seien:

3.1 MEHRRECHNER-PEARL bei Dornier

Das MEHRRECHNER-PEARL-System von Dornier ist in den Dokumenten [2], [3] und [4] beschrieben. Es handelt sich um ein Programmiersystem, das ursprünglich für Avioniksysteme konzipiert wurde und daher die erste, in Abschnitt 2 geschilderte Einsatzumgebung vorfindet. Insbesondere ist das zugrundeliegende gerätetechnische Kommunikationssystem ein paralleler Bus bei gleichzeitig loser Kopplung der Prozessoren.

Die Erweiterung zu MEHRRECHNER-PEARL besteht in dem Attribut "GLOBAL NET" (Gegenstück: "IN MODULE n") für alle netzglobalen Programmobjekte (Variable, SEMAphore, TASKs u.a.m.). Hierdurch werden Compiler und Binder veranlaßt, diese Objekte in ein spezielles Kommunikationsverfahren einzubetten und im Bedarfsfall die entsprechend geänderten Operationen auf solche netzglobalen Objekte zu generieren. Das PEARL-Ablaufsystem besteht aus einem Prozeßumschalter als Kern (ca. 300 Worte) und Bibliotheksprozeduren, die bei Bedarf vom Binder eingefügt werden, wenn ein PEARL-MODULE gebunden wird. Dadurch wird das Betriebssystem jeweils anwendungsspezifisch generiert. Die Kommunikation wickelt ein als Bibliotheksprozedur verfügbares Treiberprogramm ab.

Das Programmerzeugungssystem ist ablauffähig auf den Rechnern PDP-11/70, VAX 750, AEG 80-20 und SIEMENS 7.760. Es erzeugt Code für die Zielsysteme MUDAS, AEG 80-20 und INTEL 8086. Ein Testsystem, lauffähig auf dem Zielsystem MUDAS, erlaubt über die Compilerausgabe (hexadezimale Adressen) das Setzen von Haltepunkten sowie Lesen und Schreiben von Speicherzellen, ebenfalls hexadezimal. Für das Zielsystem AEG 80-20 ist ein in PEARL realisiertes Testsystem auf Quellsprachebene vorgesehen.

3.2 MEHRRECHNER-PEARL bei IITB

Der Sprachumfang des IITB-Compilers [6] erfüllt Basic-PEARL nach DIN 66253 vollständig und bleibt bei den darüber hinausgehenden Sprachelementen innerhalb von Full-PEARL [5]. Diese Sprachelemente sind u.a. TYPE, der Datentyp REF, die Dateibehandlung und vollständige formatierte Standard-Ein-/Ausgabe, der CHARACTER-Datentyp sowie die Feldinitialisierung.

Die Erweiterungen zu MEHRRECHNER-PEARL sind in [7] konzipiert und ihr Realisierungsweg in [8] beschrieben. Es handelt sich um den strukturbeschreibenden STATIONS-Teil, den LOAD-Teil für die Programm-Konfiguration und -Rekonfiguration sowie eine Erweiterung des SYSTEM-Teils für alternative Datenwege und Ersatzwerte; im eigentlichen PEARL nach DIN 66253 wurde nichts geändert.

Aufgrund der hohen Flexibilitätsanforderungen ist jeder Rechner mit einem vollständigen PEARL-Betriebssystem für alle Tasking-Funktionen und die (Uhr-) Zeitverwaltung sowie mit einem Netzwerk-Betriebssy-

stem für die Kommunikation ausgerüstet. Zur Erhöhung der Fehlertoleranz können MODULEs dynamisch verlagert werden. Dies erfolgt durch einen dynamischen Lader, der durch Betriebs- und Fehlerzustände gesteuert wird.

Das Programmerzeugungssystem ist ablauffähig auf SIEMENS 310, 300R30 und 7.760 sowie ab 1982 auf INTEL-Entwicklungssystemen Serie III; es erzeugt Code für SIEMENS 310 und 300R30, das RDC-System des IITB sowie INTEL 8086. Ein in PEARL geschriebenes Testsystem auf Quellsprachebene für die Zielmaschinen (auch Remote-Testing für Teilrechner) ist in Arbeit. Ein Ablaufsimulator auf dem Time-Sharing-System SIEMENS 7.760 ist vorgesehen.

4. REALISIERUNG VON MEHRRECHNER-PEARL

Im folgenden sind die in Abschnitt 3 erwähnten Realisierungen von MEHRRECHNER-PEARL genauer beschrieben. Diese Realisierungen bauen grundsätzlich auf einigen sie besonders begünstigenden PEARL-Eigenschaften auf:

- Strukturierung von PEARL-Modulen in inhaltlich unterschiedliche Teile: SYSTEM-Teil und PROBLEM-Teil
- Kommunikation zwischen PEARL-Modulen über GLOBALE Objekte
- Explizite Sprachmittel zur Reaktion auf Fehlerzustände: ON-Reaktion.

Damit ist PEARL vom Entwurf her bereits "MEHRRECHNER-fähig".

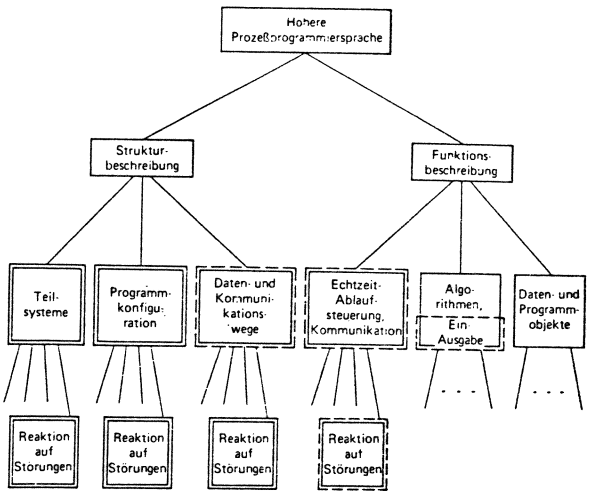

Bild 1: Grundstruktur einer Prozeßprogrammiersprache für verteilte Automatisierungssysteme.

4.1 Grundstruktur einer Prozeßprogrammiersprache für verteilte Systeme

Um die in den folgenden Abschnitten beschriebenen Erweiterungen von PEARL für den Einsatz in MEHRRECHNER-Systemen einordnen zu können, ist in Bild 1 die Grundstruktur einer Prozeßprogrammiersprache für verteilte Systeme dargestellt. Es fällt auf, daß neben den in üblichen Programmiersprachen bereits vorhandenen Sprachelementen für die Beschreibung von Funktionen ein weiterer Sprachenteil für die Beschreibung von Strukturen vorgesehen ist. Die doppelt umrahmten Funktionen sind - besonders im Hinblick auf den Einsatz für die Programmierung von Automatisierungssystemen - nur bei wenigen modernen Programmiersprachen, zum Teil nur bei PEARL verfügbar; diese Funktionen sind daher gestrichelt eingerahmt. Die mit durchgezogenen Strichen doppelt eingerahmten Funktionen sind heute in keiner Programmiersprache vorhanden; sie sind daher in MEHRRECHNER-PEARL zu ergänzen.

4.2 Die MEHRRECHNER-PEARL-Realisierung von DORNIER

Die in Literatur [2], [3] und [4] beschriebene Realisierung des MEHRRECHNER-PEARL-Systems der Firma DORNIER bezieht sich ausschließlich auf die Kommunikation in einem verteilten Mehrprozessor-System. Für diese Kommunikation kann der Programmierer sogenannte netzglobale Größen definieren (Bild 2), indem er hinter das GLOBAL-Attribut das Zusatz-Attribut NET fügt. Der Compiler wird dadurch veranlaßt, den Aufruf einer speziellen Netz-Kommunikationsprozedur abzusetzen. Diese Kommunikationsprozedur teilt Veränderungen netzglobaler Größen im gesamten Netz mit. Eine Ausnahme bilden TASKs; sie werden ebenfalls mit dem Attribut GLOBAL NET deklariert, bei ihrer Spezifikation wird jedoch explizit angegeben, in welchem Modul die TASK deklariert wurde.

```
MEHRRECHNER - PEARL für "EMBEDDED SYSTEMS" (DORNIER)
-----------------------------------------------------
Kommunikation über  n e t z g l o b a l e Größen :
■ Allgemeine Objekte
                DCL F (10) FIXED (15) GLOBAL NET ;
                SPC VALUE  FLOAT (31) GLOBAL NET ;

■ SEMAphore
                DCL A SEMA GLOBAL NET ;
                SPC A SEMA GLOBAL NET ;

■ TASKs
                T : TASK PRIO 3 GLOBAL NET ;
                SPC T TASK IN MODULE 11 ;

Keine Initialisierung von netzglobalen Größen !
Keine netzglobalen Prozedurparameter mit IDENT
J e d e  Wertänderung netzglob.Obj. systemweit
```

Bild 2: MEHRRECHNER-PEARL-Ansatz der Firma Dornier.

Da das MEHRRECHNER-PEARL-System von DORNIER eine lose Kopplung der Prozesse voraussetzt, ist die Verwendung netzglobaler Größen als Prozedurparameter mit dem Attribut IDENT nicht vorgesehen. Auch können netzglobale Größen nicht initialisiert werden, da sie in mehreren Kopien im Kommunikationssystem existieren und daher bei Systemanlauf eine dynamische Initialisierung vorgesehen werden müßte.

Die in Bild 2 gezeigten Sprachmittel für die Kommunikation über netzglobale Größen erweiteren den PEARL-Standard nach DIN 66253.

4.3 MEHRRECHNER-PEARL-Realisierung des IITB

Die MEHRRECHNER-PEARL-Realisierung des IITB berücksichtigt alle in Bild 1 doppelt umrahmten Komponenten einer Prozeßprogrammiersprache für verteilte Systeme. Die Spracherweiterungen sind so gestaltet, daß PEARL nach DIN 66253 nicht verändert wird. Die Erweiterungen betreffen insbesondere die Teilsystem-Beschreibung in einem eigenständigen Sprachenteil, die Beschreibung der Programmkonfiguration und -Rekonfiguration in einem zusätzlichen sogenannten Lade-Teil innerhalb von PEARL-Modulen sowie eine Erweiterung des PEARL-SYSTEM-Teils um die dynamische Zuordnung von Prozeßdaten-Ersatzwegen und -Ersatzwerten bei Störungen. Dieser Ladeteil sowie die Erweiterung des PEARL-SYSTEM-Teils sei im folgenden kurz beschrieben.

4.3.1 Programmkonfiguration und -Rekonfiguration

Sprachelemente zur Beschreibung einer Programmkonfiguration und -Rekonfiguration in Abhängigkeit vom Betriebszustand des Automatisierungssystems seien Ladeanweisungen (Bild 3) genannt. Sie bestehen aus der Angabe eines Teilsystems, aus einer sogenannten Ladepriorität, aus einer Betriebszustandsbedingung und aus optionalen Zusätzen. Die Ladepriorität gibt die Wichtigkeit eines Programmoduls relativ zu anderen Moduln innerhalb eines Teilsystems an (Funktionspriorität), sie ist zu unterscheiden von der Ablaufpriorität einzelner Teilprogramme. Der Betriebszustand INITIAL bezeichnet den Urlade- und Normalzustand, während die in Klammern eingeschlossenen logischen Ausdrücke nicht normale Betriebszustände mit Hilfe von Statuszeichnern aus der Teilsystembeschreibung beschreiben. Die optionale Angabe einer Startnummer für den Initialzustand beschreibt die Urstart-Reihenfolge eines Programmsystems. Das optionale Attribut RES[IDENT] bewirkt ein Urladen von Programmen in Ersatz-Teilsysteme, um dort auch bei Ausfall der Programmladeeinrichtung verfügbar zu sein oder bei zeitkritischen Programm-Rekonfigurationsfällen den Ausfall von Verarbeitungsfunktionen zu verkürzen.

```
MODULE (MOD1);

LOAD;
TO STA1 LDPRIO 5 INITIAL STARTNO 1;
TO STA3 LDPRIO 5 ON (STA1PR AND NOT STA3PR) RES;
TO STA2 LDPRIO 5 ON (STA1PR AND STA3PR);
   :
SYSTEM;
   :     /* DESCRIPTION OF DATAWAYS, ADAPTATION OF
   :        DATA ITEMS, REPLACEMENT DATA          */
   :
PROBLEM;
   :
   :     /* ALGORITHMS, PROGRAM FLOW CONTROL, I/O  */
   :
MODEND;
```

Bild 3: MEHRRECHNER-PEARL-Ansatz des IITB: Modul-Ladeteil.

4.3.2 Prozeßdaten-Ersatzwege und -Ersatzwerte

Die Sprachelemente zur Beschreibung von Prozeßdaten-Ersatzwegen und -Ersatzwerten bei Störungen sind in einer Datenwegbeschreibung (PEARL-SYSTEM-Teil) angeordnet, um die Beschreibung von Datentransport und Datengewinnung von den Verarbeitungsalgorithmen zu trennen (Bild 4).

```
SYSTEM;
  :
  :
TEMP09 : ->  ANIN560 * 5

 /*$     ->  ANIN571 * 5,
            CORR: TEMP09=TEMP09/3-16

        ->  ANIN572 * 5,
            CORR : CALL ADJUST (TEMP09,2)

        ->    REP   : TEMP09 = 1300,

              PLAUS: (HI=1500,LO=1000,DELTA=10)*/;

HEAT09 : <- ANOUT560 * 15 ..............  ;
  :
  :
```

Bild 4: MEHRRECHNER-PEARL-Ansatz des IITB: Prozeßdaten-Ersatzwege und -Ersatzwerte.

Der Name TEMP einer Prozeßdatenquelle möge eine einzulesende Temperatur bezeichnen, deren Meßfühler standardmäßig mit dem Gerät ANIN560, Kanal 8, einem Analogeingang, verbunden sei. Alternative Verbindungen sollen bestehen zu den Geräten ANIN571 und ANIN572, gegebenenfalls auch von zusätzlichen (redundanten) Meßfühlern her. Bei diesen Alternativen können Korrekturalgorithmen zur Anpassung der eingelesenen Werte an unterschiedliche Geräte- und Datenwegeigenschaften angegeben werden. Die letzte Alternative bezeichnet einen Ersatzwert, in derselben Zeile stehen auch die Prüfkriterien für Plausibilitäts-

prüfungen der gewonnenen Meßwerte. Die Bearbeitung einer solchen Datenwegbeschreibung erfolgt in der Reihenfolge ihrer Notation, abhängig vom Betriebszustand der Datenwege und dem Ergebnis der Plausibilitätskontrolle. Sind alle Datenwege gestört oder liefern sie nicht plausible Werte, so gilt der Ersatzwert als Meßgröße. Damit steht für die mit Prozeß-Ein/Ausgabedaten korrespondierenden Variablen der algorithmischen Programmteile (PEARL-PROBLEM-Teil) immer ein gültiger Wert zur Verfügung, ohne die Verarbeitungsalgorithmen mit Reaktionen auf Störungen zu belasten.

5. ANFORDERUNGEN AN MEHRRECHNER-PEARL IM VERTEIDIGUNGSBEREICH

Die vorstehend beschriebenen Realisierungen von MEHRRECHNER-PEARL sind erprobte und eingesetzte Entwicklungen für die in Abschnitt 2 dargestellten Einsatzumgebungen. Da diese Einsatzumgebungen jeweils Extreme darstellen, ist

> ein funktional abgestufter Satz von Ergänzungen zu PEARL für den Einsatz in MEHRRECHNER-Systemen unterschiedlichen Typs

anzustreben. Ein solcher Satz von Ergänzungen, aufgebaut auf die Ergebnisse und Erfahrungen mit den bisher vorhandenen und noch durchzuführenden Implementationen und Anwendungen, sollte durch die Normung von MEHRRECHNER-PEARL in seiner Einsatzfähigkeit unterstützt werden.

Ebenso wichtig ist aber auch die Unterstützung durch angepaßte Programmierumgebungen. Hier sind vordringlich Testsysteme auf Hochsprachenebene unter Berücksichtigung der speziellen Probleme von MEHRRECHNER-Systemen erforderlich. Insbesondere ist das Testen von Programmen in verteilten Prozessorstationen von einer zentralen Teststation aus zu ermöglichen. Solche Testsysteme, die auch in der Betriebsphase von MEHRRECHNER-PEARL-Programmsystemen für Wartungszwekke von hoher Wichtigkeit sind, nutzen das Kommunikationsmedium des verteilten Rechnersystems; sie müssen darüberhinaus Zugriff zu den lokalen Betriebssystemen der verteilten Mikroprozessorstationen haben. Damit liegt es nahe, neben den üblichen Programmtestfunktionen auch Statusinformationen des verteilten Rechnersystems am Testplatz verfügbar zu machen und damit die Wartung zu unterstützen.

6. Literatur

[1] DIN 66253: Programmiersprache PEARL
Deutsches Institut für Normung (DIN), DIN 66253, Teil 1 (1979) und Teil 2 (1980).

[2] Lang, W.: Sprachbeschreibung DS-PEARL Subset. Dornier System GmbH, Friedrichshafen, Nov. 1979.

[3] Blank, J. et. al.: Bericht zum Projekt "MULTICOMPUTER PEARL". Dornier System GmbH, Friedrichshafen, November 1979.

[4] Graf, H.; Lewandowski, Fr.: Bericht zum Projekt "Testsystem AEG 80/20 MIL". Dornier System GmbH, Friedrichshafen, 1982.

[5] Kappatsch, A. et. al.: Full PEARL Language Description. Kernforschungszentrum Karlsruhe (FRG). Report KFK PDV 130 (1977) und DIN 66253: Programmiersprache PEARL, Full PEARL. Entwurf zum Teil 2 von DIN 66253, NI-5.8/4-80, NI-5.8.1/1.80, Deutsches Institut für Normung (DIN).

[6] Werum, W.; Windauer, A.: PEARL, Beschreibung mit Anwendungsbeispielen. Vieweg (1978).

[7] Steusloff, H. U.: Zur Programmierung von räumlich verteilten, dezentralen Prozeßrechnersystemen. Dissertation. Fakultät für Informatik der Universität (TH) Karlsruhe (1977).

[8] Heger, D.; Steusloff, H.U.; Syrbe, M.: Echtzeitrechnersystem mit verteilten Mikroprozessoren. Bundesministerium für Forschung und Technologie, Forschungsbericht DV 79-01 Datenverarbeitung (1979).