PEARL auf einer 16-Bit-Rechenanlage mit Mehrbenutzersystem

Autor: Dipl.-Ing. Bleicke Boysen, Achim

1. EINLEITUNG

Seit 1978 steht bei KRUPP ATLAS-ELEKTRONIK BREMEN ein PEARL-Programmiersystem zur Verfügung. Es wurde auf dem EPR 1100 implementiert, einem Kleinrechner mit 16-Bit Wortbreite und einer Ausbaufähigkeit auf 64 K Worte Arbeitsspeicher.

Bei der Entwicklung der Rechner EPR 1300/1400/1500 wurden die auf dem EPR 1100 gemachten Erfahrungen berücksichtigt und einige Operationen in das Mikroprogramm der Zentraleinheit aufgenommen, die sich an den Erfordernissen höherer Sprachen orientiert; z.B.: Semaphor-, Signal-, Tasking-, Zeit-, Index- und Testhilfeoperationen.

Des weiteren wurde bei der Entwicklung des Multi-User-Operating-Systems (MOS) der Umstand berücksichtigt, daß ein PEARL-System ungestört neben Benutzern laufen soll, die Editieren, Compilieren oder Testen bzw. daß auf verschiedenen Benutzern PEARL-Systeme unabhängig voneinander laufen oder getestet werden. Aber auch der Gesichtspunkt, daß der Rechner mit seiner großen Ausbaufähigkeit einen Prozeß bearbeiten soll, mußte berücksichtigt werden .

2. PEARL AUF MEHREREN BENUTZERN

Integrationsprobleme, deren Ursachen darin liegen, daß Code- und Datenmenge den adressierbaren Raum übersteigen, sind bei grösseren Projekten keine Seltenheit. Dies um so mehr, je größer die Zahl der Mitarbeiter oder Gruppen ist, die an der Realisierung eines Projektes beteiligt sind, und je schlechter die Kommunikation unter den Beteiligten ist. Die in PEARL angebotene Möglichkeit, Code und Daten nicht-resident zu fahren, kann nur dort zur Lösung beitragen, wo die Differenz zwischen Programmgröße und adressierbarem Raum dies noch zuläßt, und wo die damit einhergehende Verschlechterung der Reaktionszeiten abschätzbar und tragbar ist. Steht nun ein Rechner zur Verfügung, dessen physikalische Grenze für die Arbeitsspeicherkapazität bei 4 M Worten liegt, dessen adressierbarer Raum für einen Benutzer aber nur 64 K Worte umspannt, so scheint es geradezu widersinnig zu sein, wenn diese Begrenzung bei der Realisierung umfangreicher Projekte durch den Anwender akzeptiert werden müßte. Auch der Umstand, daß der 64 K Benutzerraum nur durch das PEARL-Laufzeitpaket eingeschränkt ist, und das Betriebssystem mit seinen Verwaltungsblöcken und Kernfunktionen diesen Raum nicht belasten, kann diese Problematik zwar entschärfen, aber nicht lösen. Die Strukturierungsmöglichkeiten in PEARL und das Konzept der Parallelverarbeitung legen bei einem Multi-User-Betriebssystem die Lösung nahe, PEARL auch auf mehreren Benutzern zu fahren. Im Vergleich mit den Konzepten, PEARL auf mehreren Rechnern zu fahren, beschränkt sich die Problematik in diesem Fall auf die Themen

- Synchronisation und
- Datenaustausch,

da z.B. Fehlertoleranzverhalten und Rekonfigurationsprobleme, wie sie in [1] beschrieben werden, hier unberücksichtigt bleiben können, denn das System läuft in diesem Fall auf einem Prozessor.

2.1 Synchronisation

PEARL stellt die Synchronisationsanweisungen REQUEST und RELEASE auf das Objekt SEMAPHOR

zur Verfügung. Hier kann das Verfahren, wie es für einen Benutzer realisiert wurde, leicht übertragen werden auf die Synchronisation zwischen mehreren Benutzern, denn das Betriebsmittel Zentralprozessor (CPU) wird ausschließlich nach dem Gesichtspunkt Priorität vergeben.

Da in diesem System benutzerunabhängige Semaphore generiert werden können, und der Anwender diese über die Benutzerbeschreibung leicht verfügbar machen kann, bleibt nur noch das Problem, diese Semaphore in den PEARL-Modulen ansprechen zu können und sie von dem benutzereigenen Semaphore zu trennen.

Die benutzereigenen Semaphore werden im Problemteil deklariert, während die benutzerübergreifenden Semaphore in der speziellen Compilationseinheit PRELUDE deklariert werden und im Problemteil nur noch spezifiziert werden.

Als weiteres Synchronisationshilfsmittel steht natürlich der Weg über die DATION und eine Messagebox zur Verfügung.

2.2 Benutzerübergreifende globale Daten

Bei der Parallelverarbeitung sind gemeinsame Datensätze für die Aktualisierung des Abbilds eines realen Prozesses und für den Informationsaustausch zwischen den eigenständigen Aktivitäten unabdingbar. Da in PEARL auch noch die einzelnen parallelen Tasks nicht einer Compilationseinheit zuzuordnen sind, hebt PEARL mit dem Attribut GLOBAL die Objekte konsequenterweise aus der Blockstruktur und der Compilationseinheit heraus und macht sie allgemein verfügbar. Betrachten wir jetzt die globalen Daten in einem auf mehrere Benutzer verteilten PEARL-System, so ist es wohl denkbar, daß diese Daten den Adressräumen der Benutzer zugeordnet werden, für die sie deklariert wurden.

Nun gibt es aber in einem Multi-User-System Schutzmechanismen, die es nicht zulassen, in dem Arbeitsspeicher eines anderen Benutzers zu agieren. Der Datenfluß müßte also über einen Monitor kanalisiert werden, der Legalitätsprüfungen vornimmt. Das wiederum bedeutet, daß Kontrollblöcke zu diesen globalen Daten angelegt werden müßten, und in Summe neben einem Vielfachen an Zeit auch noch der Benutzerraum durch zusätzliche Kontrollblöcke und Monitor-Calls belastet würde.

Um all diese Verluste zu vermeiden, und um direkt auf die globalen Objekte zugreifen zu können, wurden eigene Datenräume geschaffen, die benutzerunabhängig im physikalischen Arbeitsspeicher nach Bedarf angelegt werden. Sie schränken den adressierbaren Raum eines Benutzers nicht ein und können Datenmengen bis zu 64 K Worte aufnehmen. Es können 32 solcher Datenräume angelegt werden, was bei einer maximalen Auslastung 2 M Worten entspricht. Der nützliche Nebeneffekt dieser Lösung ist, daß große Datenmengen aus dem Adressraum eines Benutzers herausgenommen werden können, und dieser Platz nun wieder zur Codierung zur Verfügung steht. Da die Adressräume der Datensegmente und der Benutzer disjunktiv sind, mußte darauf verzichtet werden, auch globale Prozeduren und Tasks in diese benutzerunabhängigen Adressräume einzulagern, da die Zentraleinheit nur den Code in Benutzeradressräumen ausführen kann.

Die Syntax von PEARL wurde in einem Punkt erweitert. Es muß zur Deklaration solcher Datenräume der Modulzeile das Attribut GLOBAL beigefügt werden, z.B.:

```
MODULE   DATA   GLOBAL   (n);
```

dabei ist n eine ganze Zahl zwischen 0 und 31 und bezeichnet das Datensegment. Alle Objekte, die in diesem Modul deklariert werden, sind benutzerunabhängig. Sollen nun verschiedene Benutzer auf diese Daten zugreifen können, so muß dieses Modul in die verschiedenen PEARL-Anwenderprogramme virtuell eingebunden werden. Dies hat keinen Einfluß auf die globalen Objekte, die in anderen Moduln deklariert wurden, und die ihren globalen Charakter benutzerspezifisch behalten.

2.3 Laufzeitverhalten

Betrachten wir zunächst die Synchronisationsmechanismen.

Bei den Semaphoroperationen REQUEST und RELEASE auf ein benutzerübergreifendes Semaphor kann der zusätzliche Zeitaufwand gegenüber den Operationen auf ein benutzereigenes Semaphor vernachlässigt werden: Er liegt bei etwa einem Speicherzyklus. Aber für ein RELEASE muß die Zeit, die im Umfeld dieser Operation verbraucht wird, noch berücksichtigt werden. Diese Operation kann bei einem verteilten PEARL-Prozess zu einem Benutzerwechsel führen, und der kostet etwa 200 µsec.

Die Synchronisationsmöglichkeit über eine Messagebox kann aber zeitlich dazu keine Alternative sein, da der Umweg über ein Dation-I/O viel Zeit kostet, der Benutzerwechsel durch das System aber trotzdem stattfinden muß.

Auch der direkte Zugriff auf benutzerübergreifende globale Daten ist nicht ganz kostenlos. Er wird zwar in diesem System durch die Firmware unterstützt, ist aber mit einer Grundlast von 14 µsec behaftet. Dagegen beträgt aber der proportionale Anteil je Wort nur 1,4 µsec. Durch dieses Verhältnis wird der Schwerpunkt der Anwendung deutlich: Bei der Zuweisung größerer Datenmengen, z.B. ganzer Strukturen, kann die Grundlast vernachlässigt werden, wird aber z.B. ein Index in ein solches Datensegment gelgt, so kann das in einer REPEAT-Schleife zu einem spürbaren zeitlichen Mehraufwand führen.

Dies soll zeigen, daß die erweiterten Möglichkeiten ein Hilfsmittel bei der Planung, Aufgabenverteilung und Strukturierung grösserer Problemstellungen sind und in der Phase der Codierung nicht ohne Bedacht eingesetzt werden sollen.

2.4 Testhilfen

Für umfangreiche PEARL-Systeme können Testhilfsmittel leicht unbrauchbar werden, wenn bei deren Benutzung der Adressraum des Benutzers eingeschränkt wird oder die Speicherbelegung sich durch unterschiedliche Code-Genierung verschiebt. Des weiteren muß in unserem Fall dem Umstand Beachtung geschenkt werden, daß ein PEARL-System auf verschiedene Benutzer verteilt laufen kann, und daß es Adressräume für Daten gibt, die benutzerunabhängig sind. Da dem Benutzer etwas anderes als ein quelltextbezogenes Testsystem nicht zugemutet werden kann, ergab sich eine umfangreiche Aufgabenstellung, bei deren Bewältigung auf die folgenden Punkte besonderer Wert gelegt wurde:

- Keine Belastung des Benutzerraumes durch den Testmonitor oder durch Namenslisten bzw. Adressbücher
- Kein unterschiedlicher Code für den Testfall oder den ungetesteten Fall. (In beiden Fällen läuft der optimale Code)
- Die Anzahl der Benutzer, die getestet werden, darf nicht durch einen vorgegebenen Parameter beschränkt sein
- Bei einem Halt in einem Haltepunkt müssen die anderen PEARL-Benutzer der definierten Testumgebung ebenfalls angehalten werden, um die Synchronität der Aktivitäten zu wahren
- Der Zustand eines PEARL-Prozesses muß beurteilt werden können, auch bei Verteilung auf mehrere Benutzer (z.B. Deadlock durch Semaphorbelegung oder durch Speicherverklemmung bei nichtresidenten Prozeduren)
- Zugriff auf globale Objekte in Datensegmenten auf der Basis der Objektnamen

Der Testmonitor arbeitet in diesem System auf einem eigenständigen Benutzer.

3. EINSATZMÖGLICHKEITEN

Daß umfangreiche Probleme, wie sie in der Mehrzahl durch die Aufgabenstellung der Industrie entstehen, auch auf einem 16-Bit Rechner gelöst werden können, zeigt die Konzeption, die in dem vorangegangenen Kapitel erläutert wurde. Dabei ist noch anzumerken, daß auch Benutzer mit in META (Systemsprache

bei KAE) gelösten Teilproblemen in ein solches System eingebunden werden können, da auch sie auf die globalen Objekte zugreifen können und sich über globale Semaphoren synchronisieren können. Ein solches Herauslösen von spezifischen Aufgabenstellungen hat auch den Vorteil, daß der Benutzer sich nicht um spezielle Schnittstellenprobleme zu kümmern braucht, und die Laufzeitumgebung durch die Benutzerkonfiguration vorgegeben werden kann.

Neben der Einsatzmöglichkeit, die im Vorangegangenen angedeutet wurde, ergeben sich vielfältige Anwendungsmöglichkeiten durch die Variationsbreite, die bei der Einrichtung von Benutzern besteht. So können auch mehrere voneinander unabhängige PEARL-Prozesse gefahren werden, die in ihrem Prioritätenspektrum durch die Benutzerbeschreibungen aufeinander abgestimmt wurden, und daneben kann getestet, editiert und compiliert werden, ohne die laufenden PEARL-Prozesse zu stören.

Zum Schluß sei noch darauf hingewiesen, daß die Möglichkeit, auch mehrere Rechner einzusetzen und die Synchronisation und den Datenaustausch über die DATION zu realisieren, durch den oben erläuterten Teilaspekt dieser PEARL-Implementation in keiner Weise beschnitten wird.

4. LITERATUR

[1] Steusloff, H.: Mehrrechner-PEARL
PEARL-Rundschau 1980, Heft 4;
PEARL-Verein, Düsseldorf