PEARL-News

Ausgabe 2 November 2001

Mitteilungen der GI-Fachgruppe 4.4.2 Echtzeitprogrammierung PEARL

ISSN 1437-5966

Impressum

Herausgeber	GI-Fachgruppe 4.4.2 Echtzeitprogrammierung PEARL URL: http://www.real-time.de
Sprecherin	Prof. Dr.-Ing. B. Vogel-Heuser Bergische Universität Fachbereich 13 Lehrstuhl Automatisierungstechnik / Prozeßinformatik Fuhlrottstraße 10 D-42097 Wuppertal Telefon: 0202/439-2945 Telefax: 0202/429-2944 E-Mail: bvogel@uni-wuppertal.de
Stellvertreter	Dr. P. Holleczek Universität Erlangen-Nürnberg Regionales Rechenzentrum Martensstraße 1 D-91058 Erlangen Telefon: 09131/85-7817 Telefax: 09131/30 29 41 E-Mail: holleczek@rrze.uni-erlangen.de
Redaktion	Prof. Dr. Dr. W. A. Halang FernUniversität Fachbereich Elektrotechnik und Informationstechnik Lehrstuhl für Informationstechnik, insb. Realzeitsysteme D-58084 Hagen Telefon: 02331/987-372 Telefax: 02331/987-375 E-Mail: wolfgang.halang@fernuni-hagen.de
ISSN	1437-5966

Redaktionell abgeschlossen am 12. November 2001

Einreichung von Beiträgen: Beiträge zu dieser Zeitschrift sind jederzeit hoch willkommen. Je früher und „rechtzeitiger“ sie eintreffen und je humorvoller sie sind, umso glücklicher ist der Redakteur. Einreichungen „auf den letzten Drücker“ und das Textverarbeitungsprogramm Word verleiten ihn hingegen zu Wutausbrüchen. Darum bittet er höflich um Übersendung der Beiträge per elektronischer Post in reinem ASCII und nicht codiert – eben ohne jeden Firlefanz – oder noch besser in LaTeX.

Inhalt

1 Neuer Arbeitskreis zur systematischen Software-Entwicklung

Software für Automatisierungs- und eingebettete Systeme wird zunehmend auch modellgestützt entwickelt. Hierbei sind wesentliche Defizite vorzufinden, insbesondere bei der Abbildung von Echtzeitaspekten. Klassische Konzepte wie SA/RT oder SAPP/PASS bieten entsprechende Konstrukte. Dagegen haben objektorientierte Modellierungsansätze wie UML noch eine Vielzahl von Schwachstellen, entwickeln sich aber zum Standard. Für die industrielle Anwendung bieten informatikorientierte Modellierungsansätze oft nur unzureichende pragmatische Unterstützung. Vor diesem Hintergrund beabsichtigt der am 10. Oktober 2001 gegründete und mit Teilnehmern aus Universität und Industrie besetzte neue Arbeitskreis der Fachgruppe sich mit systematischer Software-Entwicklung und Modellierung für Systeme mit Echtzeitanforderungen in der Automatisierungstechnik zu beschäftigen.

Der Arbeitskreis legt sein Hauptaugenmerk auf objektorientierte Programmiermethoden, die sich für die Erstellung von Software mehr und mehr durchsetzen und entsprechend weite Verbreitung finden. Der Vorteil dieser Methoden liegt in der Dokumentation der zu erstellenden Applikationen. Nicht zuletzt wächst die Auswahl unterstützender Werkzeuge stark an. Inzwischen sind auch Echtzeitvarianten verfügbar. Der Arbeitskreis beabsichtigt nicht, den Problemraum auf gewisse Branchen oder Geräte/Anlagen zu beschränken. Es ist die Frage zu klären, ob sich diese Methoden der Software-Erstellung auch zur Programmierung automatisierungstechnischer Anwendungen eignen. Vielfach bieten die Werkzeuge keine speziellen Sprachkonstrukte an, um Aufgaben der Automatisierungstechnik zu modellieren. Das Kernthema des Arbeitskreises soll die Implementierung der objektorientierten Modellierungsmethode sein. Nach genauer Definition des Objektbegriffes und Untersuchung der vorhanden Beschreibungsmethoden soll, unter Einwirkung praktischer Erfahrungen aus der Industrie, eine Vorgehensweise extrahiert werden, die zu einem Entwurfsmuster auf Grundlage der UML oder einer neu zu schaffenden Notation führt. Hierbei sollen objektorientierte Methoden mit den funktionsorientierten Ansätzen zur Modellierung von Echtzeitproblemen verglichen werden. Es ist eine Hardware-Anbindung im Sinne einer ganzheitlichen Betrachtung zu schaffen. Besonderes Augenmerk gilt hierbei nebenläufigen Prozessen.

In der ersten Arbeitsphase soll die Definition des Objektbegriffes konkretisiert werden. Dies erfolgt vor dem Hintergrund der Erfahrungen mit der funktionsorientierten Modellierung. Es wird eine Analyse der Kooperation aktiver Objekte im Hinblick auf Nebenläufigkeiten und Zeitverhalten erstellt werden. Desweiteren sollen unter Leistungsgesichtspunkten aus realen Anwendungen und Systemen Implementierungsrandbedingungen gesammelt werden. (Zur Mitarbeit an diesem Punkt werden noch weitere Interessenten aus der Wirtschaft gewünscht.) Ferner ist die Frage der Notation zur Dokumentation von Ergebnissen zu klären – kann UML mit einer speziellen Semantik ausgestattet werden oder liegt es näher, einen Neuentwurf zu initiieren? In der zweiten Phase wird der Begriff des Objektes dem des Funktionsbausteins gegenübergestellt werden. Es soll das Objekt als Architekturelement im Sinne der Implementierung untersucht werden. Die dritte Phase umfaßt die Behandlung von Fehlern und Ausnahmen sowie die Erstellung eines Implementierungsentwurfsmusters im Sinne eines Katalogs (oder einer Bibliothek). Parallel zu diesen Aktivitäten werden ständig Erfahrungen zu den einzelnen Werkzeugen sowie den Standardisierungsbestrebungen für UML ausgetauscht werden. Die Arbeitsgruppen werden sich per E-Mail austauschen und Arbeitsgruppentreffen einleiten. Zu den Treffen des Arbeitskreises werden Vorträge vorbereitet. Es werden noch weitere Industriepartner gesucht, um einen besseren Erfahrungsaustausch mit der industriellen Anwendung zu gewährleisten.

Die nächste Sitzung des Arbeitskreises wird am 24. Januar 2002 um 10.30 Uhr im Institut von Herrn Prof. Kaltenhäuser in Hamburg stattfinden. In ihr werden Arbeitsgruppen ihre Zwischenergebnisse berichten. Anläßlich des Workshops Echtzeitprogrammierung 2001 wird ein erster informeller Informationsaustausch zwischen den in der FG 4.4.2 vertretenen Mitgliedern des Arbeitskreises stattfinden.

An der der Mitarbeit in diesem Arbeitskreis Interessierte wenden sich bitte an:

Prof. Dr.-Ing. Birgit Vogel-Heuser
E-Mail: bvogel@uni-wuppertal.de
Tel.: 0202/439-2945

Dipl.-Ing. Martin Rüffer
E-Mail: mrueffer@uni-wuppertal.de
Tel.: 0202/439-3158

Bergische Universität
Fachbereich Elektrotechnik und Informationstechnik
Lehrstuhl für Automatisierungstechnik / Prozeßinformatik
Fuhlrottstraße 10
D-42119 Wuppertal
Fax: 0202/429-2944

2 PEARL im Echtzeitsystemlabor der Universität Maribor

Das Labor für Echtzeitsysteme der Falkultät für Elektrotechnik und Informatik an der Universität Maribor in Slowenien ist dem Thema Realzeitsysteme in Forschung und Lehre gewidmet. Obwohl das Laboratorium erst vor drei Jahren formell gegründet wurde, beschäftigt sich die es tragende Arbeitsgruppe bereits seit eineinhalb Jahrzehnten mit dem Entwurf von Echtzeitsystemen.

Seit etwa zehn Jahren werden Kurse zu verschiedenen Themen aus dem Bereich Echtzeitsysteme im Grund- und Hauptstudium sowie auf postgradualem Niveau angeboten. Dabei wird von Anfang an für Programmierübungen die Sprache PEARL mit dem Ziel eingesetzt, Multitasking unter besonderer Betonung des Zeitaspektes in der Programmierung zu lehren.

Zu diesem Zweck wurde der frei verfügbare und unter Linux laufende PEARL-Compiler der Firma Werum mehrere Jahre lang bis zum Sommersemester 2001 zu voller Zufriedenheit aller Beteiligten verwendet. Um den Entwurf vollständiger und Schnittstellen zur Prozeßperipherie umfassender eingebetteter Systeme zu ermöglichen, wurde das Labor in diesem Sommer mit einer Anzahl auf dem Prozessor Motorola 68070 aufbauender Einplatinenrechner der FernUniversität Hagen ausgestattet, auf denen das Betriebssystem RTOS-UH und die PEARL-Entwicklungsumgebung der Universität Hannover installiert sind. Um den Bestand an Experimentierplattformen zu erweitern, wird die Anschaffung weiterer Peripherieschnittstellen und Prozeßmodelle erwogen.

Im Bereich der Forschung ist die Arbeit des Labors sehr stark durch mehr als zehnjährige intensive Zusammenarbeit mit Herrn Prof. W. A. Halang beeinflußt, der auch Doktorvater dreier seiner Mitglieder war. Um eingebettete Systeme in konsistenter Weise zu entwerfen, wurde ein ganzheitlicher Ansatz gewählt, der den Entwurf anwendungsorientierter Hardware, die Implementierung eines geeigneten Betriebssystems und die Untersuchung von Konstrukten zur Hochsprachenprogrammierung umfaßt. Diese Arbeiten resultierten im Konzept einer asymmetrischen Mehrrechnerarchitektur mit spezialisierten Prozessoren, die in Form einer Laborplattform mit vier Mikrocontrollern und verteilten Peripherieeinheiten sowie eines maßgeschneiderten Betriebssystemkerns realisiert wurde. Weiterhin wurde eine auf PEARL basierende höhere Echtzeitprogrammiersprache mit Namen miniPEARL entworfen. Es handelt sich dabei um eine Teilmenge von PEARL 90 erweitert um einige Konstrukte zur Gewährleistung zeitlich vorhersehbaren Programmablaufverhaltens auf dafür geeigneten Plattformen. Für miniPEARL wurde ein prototypischer Compiler gebaut, der einen Analysator zur Bestimmung ungünstigster Programmausführungspfade und -zeiten umfaßt. Um möglichst realistische Abschätzungen zu erhalten, wurde auch ein automatisiertes Meßsystem entwickelt, das durch den eingebauten „Background Debbugging Mode“ der Motorola-Mikroprozessoren unterstützt wird. Mit Hilfe dieses Systems läßt sich die Laufzeit linearer Programmsequenzen unter dem Einfluß aller Architekturmerkmale wie Pipelines oder Caches genau messen. Die so gewonnenen Werte gehen dann in die Berechnung der längsten Rechenzeiten von Tasks ein.

In den letzten Jahren konzentrierten sich die Forschungsanstrengungen des Labors auf zwei Hauptgesichtspunkte:

1. Einführung der Objektorientierung in die Programmierung eingebetteter Echtzeitsysteme auf der Basis von PEARL und
2. gemeinsamer Entwurf der Hard- und Software solcher Automatisierungssysteme auf der Grundlage von Mehrrechner-PEARL (DIN 66253 Teil 3).

Einige grundlegende Ergebnisse dieser Arbeiten wurden bereits als Zusammenfassung zweier Dissertation in den PEARL-News vorgestellt. Für Details zu obigen Ideen sei auf die Homepage des Labors verwiesen (`www.rts.uni-mb.si`), die auch Abstracts aller Veröffentlichungen der Arbeitsgruppe enthält. Interessenten werden gesamte Artikel gerne zugeschickt.

Zur Zeit nimmt das Echtzeitlabor schließlich an einem europäischen Projekt des fünften Rahmenprogrammes teil, dessen Konsortium von der Universität Duisburg geführt wird. Ziel des Projektes ist die Erhöhung der Zuverlässigkeit von Automatisierungssystemen durch dynamische Rekonfiguration von Betriebsmitteln. Im Rahmen des Projektes obliegt es dem Labor, auf der Grundlage früherer Arbeiten ein fehlertolerantes Rechnersystem zu entwickeln.

Prof. Matjaž Colnarič
Univerza v Mariboru, Tehniška Fakulteta, Smetanova ul. 17, SLO-62000 Maribor
colnaric@uni-mb.si

3 Neue Konzepte zu PEARL von der Universität Maribor

Nachdem im letzten Abschnitt das Echtzeitsystemlabor der Universität Maribor vorgestellt worden ist, sollen nun drei dort entwickelte und mit PEARL kompatible Konzepte zur Verbesserung der Verläßlichkeit von Echtzeitprogrammen umrissen werden.

3.1 Ausnahmebehandlung

Jedes im Laufe regulärer Programmausführung auftretende außergewöhnliche und Aufmerksamkeit erfordernde Ereignis verzögert diese Programmbearbeitung und macht so a priori bestimmte Laufzeitabschätzungen obsolet. Deshalb sollten solche Ausnahmen entweder vermieden oder verhindert werden. Zu diesem Zweck wurden einige pragmatische Erwägungen angestellt und beispielhafte Maßnahmen betrachtet und ihre Brauchbarkeit im Hinblick auf Implementierung in eingebetteten Echtzeitsystemen mit harten Zeitbedingungen diskutiert.

Infolge von Hardware-Ausfällen oder unentdeckt gebliebener Spezifikations- und Programmierfehler ist es jedoch immer noch möglich, daß etwas Unvorhergesehenes geschieht. Solche Fälle weder verhindernoch vermeidbarer katastrophaler Ausnahmen können in wohl strukturierter und vorhersehbarer Weise mit einem auf Wiederherstellungsblöcken mit Vor- und Nachbedingungen beruhenden Verfahren behandelt werden. Um nicht zu lang zu werden, sei hier auf Details verzichtet und bei der Darstellung eines entsprechenden PEARL erweiternden Konstrukts auf die Selbsterklärbarkeit der Sprache vertraut:

```
block ::= block_begin block_tail

block_begin ::= BEGIN
              | PROCEDURE parameters & attributes;
              | TASK parameters & attributes;
              | parameters REPEAT

block_tail ::=[declaration_sequence]
              [alternative_sequence] END;
declaration_sequence ::= block-specific declarations
              [PRESERVE global_var_list]
alternative_sequence ::=
              {[ALTERNATIVE [PRE bool-exp;] [POST bool-exp;]]
              [statement_sequence]}
```

3.2 Systemüberlastung

In dynamischen Automatisierungssystemen kann ein außergewöhnlicher Zustand auch durch exzessive Bedienungsanforderungen aus der Umgebung heraus auftreten und so Systemüberlastung hervorrufen. Theoretisch sollte es immer möglich sein, Überlastsituationen durch in der Entwurfsphase durchzuführende Zuteilbarkeitsanalysen zu verhindern. Leider wird dies in der Praxis nur selten getan. So kann es vorkommen, daß durch entsprechende externe Anforderungen Tasks häufiger als spezifiziert aktiviert werden.

Um derart hervorgerufene Systemüberlastungen nach dem Prinzip der allmählichen Leistungsabsenkung zu beheben, wurde im Rahmen der oben erwähnten Arbeiten zum konsistenten Entwurf eingebetteter Echtzeitsysteme mit dem Ziel der Erhöhung ihrer Verläßlichkeit ein Lösungsansatz entwickelt und in Form der Syntax einer Erweiterung einer objektorientierten Echtzeitprogrammiersprache formuliert. Er beruht darauf, für Tasks verschiedene Rümpfe mit unterschiedlichen Laufzeiten bereitzustellen. Die dabei erforderlichen Task-Ausführungszeiten werden für alle möglichen Ausführungspfade abgeschätzt, was die für die Zuteilbarkeitsanalyse sowohl normaler als auch kritischer Betriebszustände notwendige Grundlage darstellt.

Die vorgeschlagene Syntax ist im Folgenden angegeben. Alternativ ausführbare Prozeduren müssen denselben Namen und dieselben Parameter haben und müssen als virtuell deklariert werden. Wiederholte Deklaration der Parameter und eines möglichen Ergebnisses in den verschiedenen Alternativen erübrigt sich wegen ihrer Gleichheit. Bereits von frühen Entwurfsstadien an hält diese Syntax den Entwickler dazu

an, Alternativen für Task-Rümpfe und -Methoden zu betrachten. Durch die automatisierte Ausführungszeitanalyse ist er sich immer bewußt, zu welchen Zeiteinsparungen bestimmte Implementierungsvarianten führen.

```
TYPE
  MYTASK CLASS(TASK) [
  PROTECTED
    Main PROCEDURE OVERRIDE;
    Main ALTPROCEDURE;
  PUBLIC
    ...
    Foo PROCEDURE(x,y FIXED) RETURNS (FIXED) VIRTUAL;
    Foo ALTPROCEDURE;
    ...
  ];
```

Ein anderes nützliches Konstrukt zur realistischeren Gestaltung von Laufzeitabschätzungen stellt die `update`-Klausel dar. Während der Übersetzung berechnet und speichert der Analysator die restliche Programmlaufzeit bezogen auf die Programmstelle der Klausel. Zur Laufzeit wird dann dieser Wert durch die Differenz aus der ungünstigten Gesamtlaufzeit und der bis zum Erreichen dieser Programmstelle tatsächlich akkumulierten Ausführungszeit ersetzt. Dadurch wird i.a. der Tasks zur zeitgerechten Verarbeitung zur Verfügung stehende Spielraum erhöht. Dies gilt insbesondere nach Beendigung von Schleifen oder wenn sehr verschiedene mögliche Programmpfade wieder zusammenkommen.

3.3 Programmierung von Peripherieschnittstellen

PEARL wurde auch erweitert, um den Zugriff auf Peripherieschnittstellen zu ermöglichen. Aus der Sicht der Objektorientierung können jeder Typ einer Peripherieschnittstelle als eine Klasse und individuelle Schnittstellen als deren Instanzen betrachtet werden. Methoden solcher Klassen rufen vordefinierte Funktionen oder Prozeduren auf, die in den Peripherieschnittstellen angesiedelt sind. Sie können entweder nur auf die eigentlichen Hardware-Register zugreifen oder auch bestimmte komplexere Funktionen wahrnehmen. Aufruf auf anderen Rechnern laufender Prozeduren und/oder Parameterübergabe sind systemabhängig und werden physisch durch Systemprogramme implementiert.

In den Klassen werden neben den Methoden auch Eigenschaften in Form von Pseudovariablen spezifiziert, die sowohl tatsächliche Register von Peripherieschnittstellen darstellen aber auch implizit durch Methoden implementiert sein können. Eigenschaften können zeitgebunden sein, d.h. Zugriffsmethoden zu ihnen können auf Aktivierung zu bestimmten Zeitpunkten hin programmiert sein.

Im folgenden werden entsprechende Definitionen in Form selbsterklärender Syntax gegeben:

```
PeripheralClassSpecification ::=
  PeripheralClassName PERIPHERAL
    ['(' PredecessorPeripheralClass ')']
   '['{ PeripheralMethod | PeripheralProperty }']'

PeripheralMethod ::=
  PROCEDURE ProcName ['(' FormalParameters ')']
    [RETURNS DataType]
    [ImplementationParameters]

PeripheralProperty ::=
  PROPERTY PropName [ READ PropAccessMethod]
                    [ WRITE PropAccessMethod]

PropAccessMethod ::=
  PORT LogicalPortNo | PeripheralMethodName
    [TIMESTAMPED]
```

Das folgende Beispiel zeigt, wie einfache A/D- und D/A-Wandler spezifiziert werden:

```
TYPE
  TypeDA PERIPHERAL(GeneralPeripheral) [
    PROCEDURE Init RETURNS FIXED;
    PROCEDURE Done RETURNS FIXED;
    PROCEDURE SetValue(V FIXED, Time CLOCK);

    PROPERTY Value FIXED WRITE SetValue TIMESTAMPED;
  ];

  TypeAD PERIPHERAL(GeneralPeripheral) [
    PROCEDURE Init RETURNS FIXED;
    PROCEDURE Done RETURNS FIXED;
    PROCEDURE GetValue RETURNS FIXED;

    PROPERTY Value FIXED READ GetValue;
  ];
```

Die eigentlichen Peripheriegeräte werden dann wie folgt deklariert:

```
DECLARE
  DA TypeDA AT ADDR 123;
  AD TypeAD AT ADDR 124;
```

Ein Wert kann nun von dem A/D-Wandler erfaßt werden mit:

```
  X := AD.Value;
```

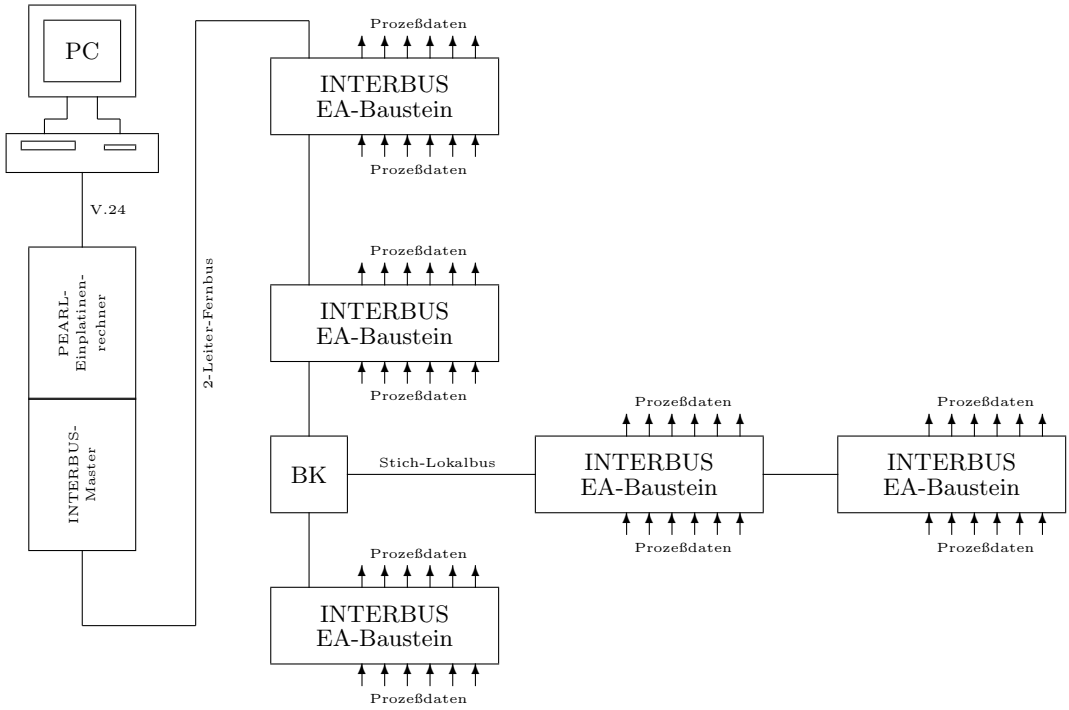
Entsprechend wird ein Wert an einen D/A-Wandler ausgegeben:

```
  DA.Value := 4 AT 12:00;
```

Prof. Matjaž Colnarič
Univerza v Mariboru, Tehniška Fakulteta, Smetanova ul. 17, SLO-62000 Maribor
colnaric@uni-mb.si

4 Realisierung eines INTERBUS-Masters mit einem PEARL-Einplatinenrechner

Sowohl das Feldbussystem INTERBUS-S der Firma Phoenix Contact als auch der Einplatinenrechner, der zum Einsatz im multimedialen Kurs „Realzeitprogrammiersprache PEARL“ der FernUniversität (vgl. Abschnitt 5) entwickelt wurde, sind durch ihre speziellen Eigenschaften besonders für harte Echtzeitanwendungen geeignet: als Buszugriffsverfahren verwendet der INTERBUS-S ein Summenrahmentelegramm, welches Daten streng deterministisch an die Feldbusteilnehmer überträgt, und der Einplatinenrechner wird mit einem der wenigen wirklichen Echtzeitbetriebssysteme (RTOS-UH) betrieben. Deshalb lag es nahe, eine Anbindung des PEARL-Einplatinenrechners an das Feldbussystem INTERBUS-S zu implementieren, um somit durch die PEARL-Orientierung des Betriebssystems eine Architektur mit präzise beschreibbarem Verhalten für die Automatisierungstechnik zu erhalten. Hard- und Software dieser Anbindung hat Herr Willi Pfeuffer im Rahmen seiner Diplomarbeit am Lehrstuhl für Informationstechnik im Fachbereich Elektro- und Informationstechnik der FernUniversität entwickelt. Das System stellt die Grundlage für den Aufbau eines Praktikumsversuches zum Thema INTERBUS dar. Die folgende Abbildung zeigt eine typische Konfiguration für ein kleines Automatisierungssystem, wie es sich mit diesen Komponenten aufbauen läßt.



Der PEARL-Einplatinenrechner ist an das Feldbussystem über den Prozeßdatenkanal des INTERBUS-S angebunden. Die Kommunikation zwischen beiden Komponenten übernimmt hierbei als zentraler Baustein der Master-Protokoll-Chip (IBS-UART) der Firma Phoenix Contact. Dies ist ein in 0,8 μm-CMOS-Technologie realisierter ASIC mit einer Komplexität von ca. 5000 Gatteräquivalenten. Der Baustein überträgt Daten mittels serieller Schnittstellen an den PEARL-Einplatinenrechner. Die maximale Übertragungsrate ist daher durch die Eigenschaften der beiden UART-Schnittstellen begrenzt. Der INTERBUS-UART bearbeitet den zeitabhängigen Teil (Medienzugriff) des INTERBUS-Protokolls, während der zeitunabhängige Teil in der Betriebs-Software des PEARL-Einplatinenrechners implementiert ist. Zu den hauptsächlichen Aufgaben des IBS-UART zählen:

- Erzeugung des Systemtaktes,
- Erzeugung der Daten- und Statustelegramme für die 2-Leiter-Schnittstelle,
- Berechnung und Kontrolle des CRC-Polynoms sowie
- Überwachung der Signale Select und Control.

Zur Kopplung des PEARL-Einplatinenrechners an den INTERBUS-S wurde eine Doppel-Layer-Platine entworfen, welche den UART-Master-Protokoll-Chip samt externer Beschaltung, Komponenten zur Erzeugung eines Rücksetzsignals und zur Ausgabe von Statusmeldungen sowie eine optionale galvanische Trennung umfaßt. Um die Kommunikation zwischen PEARL-Einplatinenrechner und INTERBUS-S auszutesten, wurden zunächst Routinen in PEARL geschrieben. Später wurden sie zur Erhöhung der Verarbeitungsgeschwindigkeit in Assembler umgesetzt.

Die entwickelte Treiber-Software ist einerseits für die reine Anwendungsprogrammierung transparent, so daß sich ein dafür geschriebenes Anwendungsprogramm nur in wenigen Punkten von einem unterscheidet, das E/A-Bausteine direkt anspricht. Andererseits kann mit Hilfe der Protokollierungsfunktionen der Ablauf des INTERBUS-Protokolls detailliert nachvollzogen werden. Die Teile der Treiber-Software, die die vom INTERBUS-UART vorgegebene Zustandsmaschine bearbeiten, sowie die Teile, die mit dem Echtzeitbetriebssystem RTOS-UH kommunizieren, sind aus Geschwindigkeitserwägungen in Maschinencode abgefaßt. Die Schnittstelle zu Anwendungsprogrammen sowie die Bedienfunktionen sind in PEARL geschrieben. Durch Verwaltung der Konfigurationsdaten in einer Baumstruktur entfällt die beim INTERBUS sonst übliche Beschränkung der Verschachtelungstiefe.

5 PEARL-Kurs und Einplatinenrechner der FernUniversität

Über den multimedialen Kurs *Realzeitprogrammiersprache PEARL* der FernUniversität wurde bereits in der letzten Ausgabe der PEARL-News berichtet, daß es gelungen ist, seinen Preis deutlich zu senken. Der Kurs umfaßt ein Lernprogramm und eine komplette Programmentwicklungsumgebung mit integrierten vollständigen Dokumentationen. Als Experimentierplattform wird ein Einplatinenrechner geliefert, der mit einem PC über eine serielle Schnittstelle kommuniziert und an den Sensoren und Aktoren, und somit beliebige Prozesse angeschlossen werden können. Vom PC aus können Prozeßautomatisierungsprogramme in den Einplatinenrechner geladen und dort unter Kontrolle des bewährten Echtzeitbetriebssystems RTOS-UH ausgeführt werden, das von Herrn Professor Gerth für diese Lernplattform freundlicherweise zur Verfügung gestellt wurde.

Wie im Abschnitt 2 erwähnt, werden dieser Kurs und der Einplatinenrechner auch von anderen Hochschulen im Rahmen von Praktika eingesetzt. Es sei darauf hingewiesen, daß der Vorrat der gefertigten Einplatinenrechner langsam zur Neige geht.

Studierende und Gasthörer der FernUniversität können den PEARL-Kurs so wie jeden anderen Kurs belegen und so CD-ROM und Einplatinenrechner beziehen. Nicht an der FernUniversität eingeschriebenen PEARL-Interessierten werden zum Zwecke der persönlichen Weiterbildung CD-ROM und Einplatinenrechner zusammen zum Preise von 173 Euro (Kursnummer 72417) und der Einplatinenrechner allein für 115 Euro (Kursnummer 72418) angeboten. Zum Bezug ist nur das von

`http://www.fernuni-hagen.de/IT/lehre/pearl.htm`

herunterladbare selbsterklärende PDF-Formular auszufüllen und einzusenden.

Wie bereits in der Ausgabe 2/2000 der PEARL-News berichtet, hat Herr Dipl.-Ing. Heinz Tatz (`Heinz.Tatz@fh-bochum.de`) die im Kurs enthaltene und auch zur Ansteuerung des Einplatinenrechners dienende Programmierumgebung überarbeitet und deutlich verbessert. Diese Umgebung wurde in der Zwischenzeit ausgiebigen Tests unterzogen und wird ab Anfang Januar 2002 unter dem oben angegebenen URL zum Herunterladen zur Verfügung stehen.