Michael Solvie

Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen

Michael Solvie

Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen

Herausgegeben von
Professor Dr.-Ing. Klaus Feldmann,
Lehrstuhl für
Fertigungsautomatisierung und Produktionssystematik

FAPS

HANSER

Carl Hanser Verlag München Wien

Als Dissertation genehmigt von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der Einreichung:	12. Juni 1995
Tag der Promotion:	18. September 1995
Dekan:	Prof. Dr. Dr. h. c. F. Durst
Berichterstatter:	Prof. Dr.-Ing. K. Feldmann
	Prof. Dr.-Ing. U. Herzog

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Solvie, Michael:
Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen / vorgelegt von Michael Solvie. - München ; Wien : Hanser, 1996
(Fertigungstechnik - Erlangen ; Bd. 52)
Zugl.: Erlangen, Univ., Diss., 1995
ISBN 3-446-18607-7
NE: GT

Herstellung: Gruner Druck GmbH, Erlangen-Eltersdorf
Printed in Germany

Vorwort und Danksagung

Die vorliegende Arbeit entstand basierend auf meiner Tätigkeit als wissenschaftlicher Mitarbeiter von Prof. Dr.-Ing. K. Feldmann am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik an der Friedrich-Alexander-Universität Erlangen-Nürnberg. Ihm gilt mein besonderer Dank für die Ermöglichung der zugrundeliegenden Arbeiten, seine großzügige Förderung und seine Anregungen. Herrn Prof. Dr.-Ing. U. Herzog danke ich für die eingehende Durchsicht der Dissertation und die Übernahme des Zweitgutachtens.

Ein herzlicher Dank gilt auch meinen Kollegen am Lehrstuhl FAPS und hier insbesondere denen aus der Gruppe für Sensor- und Steuerungstechnik für die Zusammenarbeit und die wertvolle Kritik an meiner Arbeit. Darüber hinaus gilt in fachlicher Hinsicht mein Dank Herrn Ivan Izikowitz für die zahlreichen, fachlichen anregenden Diskussionen insbesondere zum Thema *Zeit und Synchronisation* in der Frühphase der Arbeit sowie den anderen Mitarbeitern im OLCHFA-Projekt.

Für die vielen, sehr hilf- und lehrreichen Diskussionen und die Möglichkeit, die von mit bearbeitete Thematik auch aus einem anderen Blickwinkel betrachten zu können, danke ich den Kollegen aus den Teilprojekten A1, B3 und C1 des Sonderforschungsbereiches SFB 182.

Ferner gilt mein Dank den Studenten und wissenschaftlichen Hilfskräften, die mich bei der Anfertigung der Arbeit unterstützt haben. Ohne die Leistung anderer schmälern zu wollen, möchte ich an dieser Stelle namentlich die Herren cand.-ing. Angele, Dipl.-Inf. Hamadou, Dipl.-Inf. Wenzel, Dipl.-Inf. Keim, cand.-inf. Stöckel, cand.-inf. Wurm und Dipl.-Inf. Enzinger erwähnen.

Für die Durchsicht von Teilen der Arbeit und die wertvollen fachlichen Anregungen möchte ich meinen Dank an die Herren Dr. Dulz, Dr. Hofmann und Dipl.-Ing. Färber aussprechen. Für die notwendige Kritik an Formulierungen und die unermüdliche Suche nach verbuchselten Wechstaben danke ich Frau Dipl-Ing. Stief und Herrn Dipl.-Ing. Schnur sowie ganz besonders auch meinem Bruder Geert.

In privater Hinsicht gilt mein tiefster und aufrichtigster Dank meiner Familie, allen voran meiner Frau Sibylle und unseren Töchtern Julia und Laura. Sie haben durch ihre Geduld, Rücksichtnahme und Zusprache diese Arbeit möglich gemacht.

Michael Solvie

Zeitbehandlung und Multimedia–Unterstützung in Feldkommunikationssystemen

– Inhaltsverzeichnis –

Abkürzungen

ABUS	Allgemeine Bitserielle Universelle Schnittstelle
AD	Audio Device
ADPCM	Adaptive Differentielle Pulse Code Modulation
API	Application Programmers Interface
ASI	Aktuator-Sensor-Interface
ASIC	Application Specific Integrated Circuit
BKS	Buskopfstation
CAN	Controller Area Network
CCD	Charged Coupled Devices
CCIR	International Radio Consultative Committee
CIF	Common Intermediate Format
DCT	Discrete Cosinus Transformation
DDLM	Direct-Data-Link-Mapper
DPCM	Differentielle Pulse Code Modulation
DQDB	Distributed Queue Dual Bus
DQDFB	Distributed Queue Dual Field-Bus
DRSLP	Distributed Database, Realtime Services, Stream Support Link Protocol
EIA	Electronic Industries Association
ESPRIT	European Strategic Program for Research and Developement in Information Technologies
FAIS	Factory Automation Interconnection System
FDDI	Fibre Distributed Data Interface
FDL	Fieldbus Data Link
FIFO	First In First Out
FMA	Fieldbus Management
FMS	Fieldbus Message Specification
FIP	Factory Instrumentation Protocol (bzw. Flux Information Processus (frz.))
IEC	International Electrotechnical Commission
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IS	International Standard
ITU-T	International Telecommunication Union - Telecommunication Standardization Sector
JPEG	Joint Photographic Experts Group
kbps	Kilobit pro Sekunde
LAN	Local Area Network
LCO	Local Clock Object
LLC	Logical Link Control
LLI	Lower Layer Interface

Mbps	Megabit per second
MAC	Medium Access Control
MAP	Manufacturing Automation Protocol
MIB	Management Information Base
MMS	Manufacturing Message Specification
M-JPEG	Motion-JPEG
MPEG	Motion Pictures Experts Group
MPD	Motion Picture Device
MPS	Manufacturing Periodical/Aperiodical Services
MPS-E	Manufacturing Periodical/Aperiodical Services - Extended
MSDU	MAC-Service Data Unit
MSB	Most significant bit
NC	Numeric Control (Numerische Steuerung)
NRZ	Non-Return-to-Zero
NTSC	National Television Standards Committee
OLCHFA	Open Low Cost Time Critical Wireless Fieldbus Architecture
OMP	OSIRIS Management Protocol
OSI	Open Systems Interconnection
OSIRIS	Open Service Integrating Realtime fieldbus System
PAL	Phase-Alternate-Line
PC	Personal Computer
PCM	Pulse Code Modulation
PDU	Protocol Data Unit
PEARL	Process and Experiment Automation Real-Time Language
PROFIBUS	PROcess FIeldbus
QCIF	Quarter Common Intermediate Format
RC	Robot Control
RTSIMS	Real-Time Service Integrating Message Specification
SERCOS	Serial Real-Time Communication System
SAP	Service Access Point (dt. DZP)
SID	Still Image Device
SIF	Source Input Format
SNA	System Network Architecture
SPS	Speicherprogrammierbare Steuerungen
TOP	Technical and Office Protocol
TCP/IP	Transmission Control Protocol / Internet Protocol
TSO	Time Service Objekt
UART	Universal Asynchronous Receiver/Transmitter Character
VHS	Video Home System

1 Einleitung

Neben Arbeit und Kapital ist „Information" zum dritten wesentlichen Produktionsfaktor geworden [1]. Der Austausch von Informationen zwischen den datenverarbeitenden Systemen in der rechnerintegrierten Produktion gewinnt mit dem Ausmaß der Verteilung der funktionalen Komponenten an Bedeutung und die Kommunikation wird in Anbetracht des heute bereits erreichten Grades der Dezentralisierung von Automatisierungsstrukturen und im Hinblick auf dessen abzusehende zukünftige Entwicklung deshalb zu recht als eines der Schlüsselgebiete der Automatisierungstechnik bezeichnet.

In der rechnerintegrierten Fertigung wird eine durchgängige Vernetzung zwischen den verschiedenen funktionalen Bereichen der Produktion, z.B. Verwaltung, Entwicklung, Planung, Produktion und Qualitätssicherung angestrebt. Ziel ist dabei die rechtzeitige Verfügbarkeit aller relevanten Daten an u.U. verschiedenen Orten. Die zwingend damit einhergehende Verfügbarkeit aller Informationen in einer durch Rechnersysteme darstellbaren Form erleichtert die durchgängige Datenmodellierung und ist z.B. auch eine der Grundvoraussetzungen für eine Realisierung der Qualitätssicherung nach ISO 9000, der aktuell eine große Bedeutung zugemessen wird [2].

In den höheren Ebenen der Automatisierungshierachie, in denen primär planende und dispositive Aufgaben wahrgenommen werden, kommen seit geraumer Zeit in der Regel standardisierte Kommunikationssysteme mit einem breiten Anwendungsspektrum zum Einsatz. Die Ablösung der bislang in den prozeßnahen Bereichen eingesetzten parallelen Verdrahtung mit analoger Signalübertragung durch digitale Kommunikationssysteme wird mit der beginnenden Verfügbarkeit entsprechender Produkte zu akzeptablen Preisen in größerem Ausmaß möglich. Von einer stark zunehmenden Marktdurchdringung mit entsprechenden Systemen in den nächsten Jahren kann ausgegangen werden (vgl. Bild 1). Als befristete Übergangslösung auf dem Weg zum Einsatz von digitalen *Feldkommunikationssystemen* ist dabei die Kombination von analoger und digitaler Signalübertragung in einem System, die auch als *SMART*-Technologie bezeichnet wird, zu sehen.

Für den Anwender sind neben den technischen Vorteilen, z.B. der weitaus einfacheren Wartung und der Verfügbarkeit zusätzlicher Informationen, auch direkte wirtschaftliche Faktoren, insbesondere die Komponenten-, Installations- und Wartungskosten, ausschlaggebend. Auch hier weisen digitale Kommunikationssysteme für den prozeßnahen Bereich dank der preiswerten Verfüg-

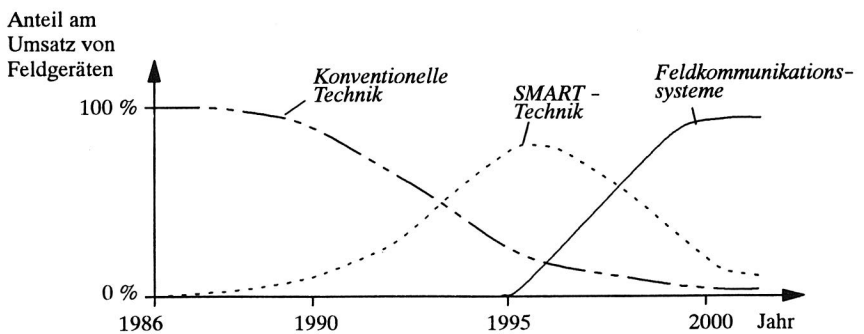


Bild 1: Rückschau und Prognose von Umsatzzahlen von Feldgeräten verschiedener Technologie (nach Schneider [3])

barkeit von integrierten Schaltkreisen, z.B. speziellen ASICs zur Unterstützung der Kommunikationsfunktionalität, inzwischen gegenüber den herkömmlichen Systemen deutliche Vorteile auf.

Mit dem Einsatz digitaler Feldkommunikationssysteme ist eine gravierende Umstellung der Steuerungs- und Leittechnik verbunden [4], so daß es letztendlich zu einer Wechselwirkung zwischen dem Einsatz derartiger prozeßnaher Kommunikationssysteme und der Weiterentwicklung der Steuerungsstrukturen kommt: Ohne die Forderung nach Dezentralisierung von Verarbeitungsleistung wäre die Entwicklung von Feldkommunikationssystemen nicht in größerem Maßstab vorangetrieben worden und mit der Verfügbarkeit derartiger Systeme eröffnen sich für den Anwender neue Möglichkeiten, z.B. die Schaffung weitgehend autonom agierender modularer Teilsysteme und die Übertragung von multimedialen Informationen. Derartige Informationen, z.B. Bewegtbilder in Kombination mit auditiven Daten, vereinfachen für den Bediener trotz der räumlichen Trennung von den technischen Prozessen deren Steuerung und Überwachung durch eine deutliche Verbesserung des Informationsangebots.

Damit gilt es, mit den Feldkommunikationssystemen als Bestandteil eines verteilten Echtzeit-Steuerungssystems (vgl. Bild 2), zukünftig zwei weiteren Forderungen zu entsprechen:

- Feldkommunikationssysteme dienen zum Austausch von Prozeßwerten, die in der Regel eine nur sehr begrenzte zeitliche Gültigkeit haben [5]. Neben der Rechtzeitigkeit des Austausches von Information wird auch die Aufnahme der Dimension „Zeit" als expliziter Bestandteil von Daten, der durch das Kommunikationssystem berücksichtigt werden muß, erforderlich, um z.B. die Synchronisation der verteilten Applikationen zu erleichtern.
- Feldkommunikationssysteme sollen Funktionen zur Unterstützung von multimedialen Applikationen, z.B. im Bereich des Bedienens und des Beobachtens, bieten. Dabei sind zum einen die besonderen Anforderungen der zu transferrierenden Daten, z.B. Stand- und Bewegtbilder oder Geräusche, und zum anderen die Steuerung der multimedialen Aufnahmegeräte zu berücksichtigen [6, 7].

Die heute eingesetzten Systeme erlauben die Berücksichtigung von Echtzeit-Anforderungen nur im Sinne der Bevorratung von Mechanismen zum periodischen Datenaustausch. In den ihnen zugrunde-

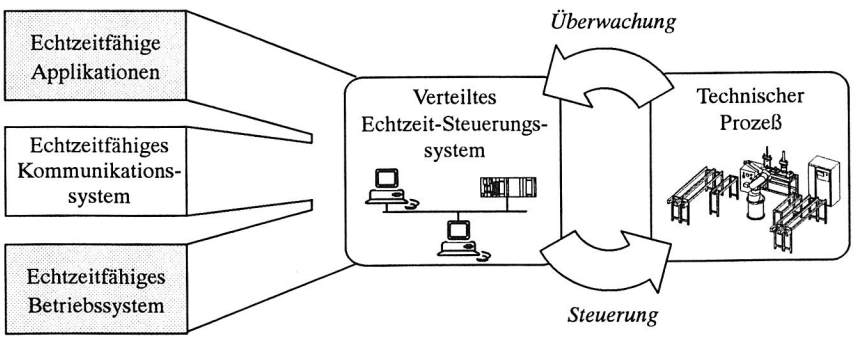


Bild 2: Echtzeitfähige Kommunikationssysteme in der Systemarchitektur eines verteilten Echtzeit-Steuerungssystems

liegenden Konzepten werden sie größtenteils nur als Verlängerung des internen Rechnerbusses der Steuerungssysteme, z.B. von speicherprogrammierbaren Steuerungen, gesehen. Eine Berücksichtigung expliziter Zeitinformationen wird nur unzureichend unterstützt. Ebenso besteht bei den verfügbaren Feldkommunikationssystemen ein Defizit bezüglich der Funktionen zur Steuerung von Systemen zur Aufnahme von Informationen die in Multimedia-Applikationen verarbeitet werden sollen (z.B. Kameras) und bezüglich der Übertragung der dort erzeugten Daten.

Diese Problematik wird in der vorliegenden Arbeit unter der Zielsetzung der Erweiterung vorhandener Systeme und der Erstellung eines Konzeptes für ein Feldkommunikationssystem, das auf die vollständige Erfüllung der Anforderungen hin ausgerichtet ist, aufgegriffen.

Die weiteren Ausführungen sind dazu wie folgt gegliedert: Im nachfolgenden Kapitel werden die notwendigen Grundlagen bezüglich der Kommunikation in dem verteilten System einer rechnerintegrierten Fertigung unter besonderer Berücksichtigung des Aspektes der „Echtzeit" vorgestellt. Dort erfolgt auch die Motivation des Einsatzes von Feldkommunikationssystemen als Basis für den Austausch multimedialer Informationen, die durch die Bereitstellung der notwendigen Grundlagen ergänzt wird.

Die daraus ableitbaren Anforderungen an Feldkommunikationssysteme sind Gegenstand der Betrachtungen im dritten Kapitel dieser Arbeit. Es wird dabei zwischen Anforderungen unterschieden, die

- primär bezüglich der Steuerung technischer Prozesse unter den Gesichtspunkten der notwendigen Echtzeitfähigkeit bestehen,
- der Unterstützung der Autonomie der Einzelsysteme als Komponenten im verteilten Steuerungssystem dienen, bzw.
- aus der geforderten Integration von Unterstützungsfunktionalität für die Übertragung multimedialer Daten erwachsen.

Im Rahmen dieses Kapitels werden mit PROFIBUS und FIP auch zwei ausgewählte, standardisierte Feldkommunikationssysteme an diesen Anforderungen gemessen und bewertet.

Ein möglicher Weg, den neuen Anforderungen zu genügen, besteht in der Erweiterung vorhandener Systeme. Diese Möglichkeit wurde unter dem Gesichtspunkt der Echtzeitfähigkeit mit Zeitbehandlung für das FIP-System verfolgt. Der PROFIBUS wurde zudem als Basis für Arbeiten genutzt, die sich mit der Nutzung dieses Systems zur Übertragung von multimedialen Daten und der Steuerung von deren Aufnahmesystemen beschäftigen. Die Vorstellung und die Bewertung der dabei erzielten Ergebnisse stellen den Inhalt des vierten Kapitels dieser Arbeit dar.

In Anbetracht der Resultate der theoretischen Untersuchungen und der Erfahrungen, die bei den praktisch durchgeführten Erweiterungen gewonnen wurden, wird – wie bereits angedeutet – deutlich, daß heutige Feldkommunikationssysteme auch in erweiterter Form nicht geeignet sind, die aufgestellten Anforderungen vollständig zu erfüllen.

Aus diesem Grunde wird in den nachfolgenden Kapiteln fünf bis sieben ein Konzept für das neue Feldkommunikationssystem *OSIRIS* vorgeschlagen, daß entlang der aufgestellten Anforderungen erarbeitet wurde. Das Konzept beinhaltet die Darstellung von Protokollen und Diensten für die Sicherungsschicht, die Anwendungsschicht sowie das Systemmanagement. Die diesbezüglichen Ausführungen werden durch eine zusammenfassende Spiegelung der Eigenschaften des Systems an den aufgestellten Anforderungen ergänzt.

2 Grundlagen und Stand der Technik

In diesem Kapitel werden die für das Verständnis der weiteren Ausführungen notwendigen Grundlagen bezüglich der Echtzeit-Kommunikation in der rechnerintegrierten Fertigung und der Multimedia-Technologie vorgestellt. Auf eine Darstellung kommunikationstechnischer Basisbegriffe, wie etwa des ISO-OSI Basisreferenzmodells, wird verzichtet. Diesbezüglich wird auf entsprechende Primär- und Sekundärliteratur (z.B. [8, 9]) verwiesen.

2.1 Struktur von Produktionssystemen

Moderne Produktionssysteme sind aus technischen und wirtschaftlichen Gründen modular aufgebaut. Das Gesamtsystem läßt sich in Teilsysteme untergliederen. Zu diesen Teilsystemen gehören u.a. die einzelnen Fertigungs- und Montagezellen, das Materialflußsystem und die Läger. Diese

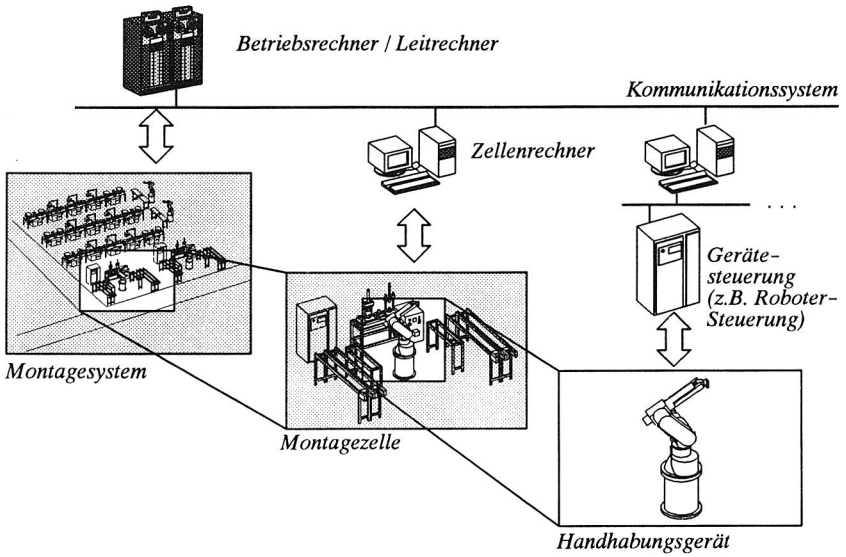


Bild 3: Modulare Struktur der rechnerintegrierten Fertigung am Beispiel einer flexiblen Montagezelle

Teilsysteme lassen sich selbst wieder in ihre Bestandteile untergliederen. In einer Montagezelle sind dies z.B. die eingesetzten Industrieroboter, NC-Achsen und das zelleninterne Materialfluß- und Handhabungssystem mit ihren, durch Universal- oder Spezialrechner gebildeten, Steuerungssystemen (vgl. Bild 3). Alle Komponenten eines Produktionssystems müssen für sich, aber auch global koordiniert im Kontext der Gesamtaufgabe gesteuert werden.

Die Dezentralisierung der Steuerungsstruktur ist dabei das Ergebnis einer Entwicklung. Die Situation in der Automatisierungstechnik anfang der 60er Jahre war geprägt durch den beginnenden Einsatz von Rechnersystemen. Die primär verwendeten Prozeßrechner wurden als funktional zentralisierte Steuerungssysteme eingesetzt. Mit dem Fortschreiten der Entwicklung der Mikroelektronik begann eine funktionale und örtliche Dezentralisierung der Steuerungsaufgaben, wobei zu-

sätzliche Steuerungssysteme, z.B. speicherprogrammierbare Steuerungen (SPS), zur autonomen Abwicklung von Teilaufgaben zum Einsatz kamen. Die Entwicklungen der letzten Jahre haben es mit der Hochintegration von Schaltkreisen und der damit einhergehenden Miniaturisierung der Bauelemente bei gleichzeitig anhaltendem Preisverfall möglich gemacht, auch die direkt am technischen Prozeß angesiedelten Systeme, z.B. Sensoren und Aktoren, mit lokaler Verarbeitungskapazität auszurüsten. Dabei ist dieser Prozeß noch nicht abgeschlossen. Es ist sowohl zu erwarten, daß der Anteil intelligenter Feldgeräte an der Gesamtzahl derartiger Systeme noch zunehmen wird, als auch, daß die Verarbeitungskapazität weiter steigen wird.

Die örtliche und funktionale Dezentralisierung birgt dabei für den Anwender eine Reihe von Vorteilen. So wird ein modularer Aufbau und eine schrittweise Inbetriebnahme der Anlagen möglich. Zudem steigen die Flexibilität und die Zuverlässigkeit des Gesamtsystems, da in dezentral aufgebauten und gesteuerten Systemen nur jeweils Teile der Anlage von dem Ausfall einzelner Komponenten betroffen sind [10, 11].

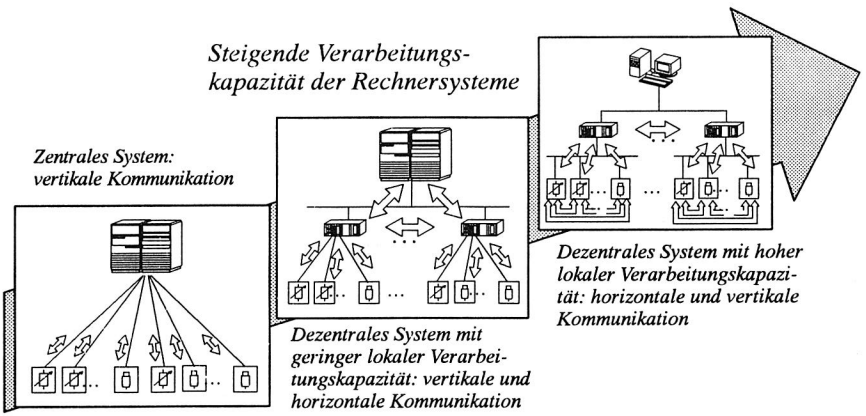


Bild 4: Entwicklung von zentralen zu dezentralen Steuerungsstrukturen

Dezentrale Steuerungssysteme sind in einem weitaus stärkeren Maße auf den Austausch von Informationen zwischen ihren verteilten Komponenten angewiesen, als zentralisierte Systeme. Dabei nimmt mit dem Grad der Dezentralisierung zum einen der Umfang des Datenaustausches zu und zum anderen ändern sich die Anforderungen an die Qualität der geforderten Kommunikationsunterstützung und an die einzusetzenden Kommunikationsstrukturen. Bei zentralen Steuerungssystemen laufen alle aus dem Prozeß aufgenommenen Informationen in einem zentralen Rechner zusammen. Es handelt sich somit um eine rein *vertikale Datenkommunikation* (vgl. Bild 4). Mit der ersten Stufe der Dezentralisierung ist die Einführung von zusätzlicher *horizontaler Datenkommunikation* verbunden. Darunter wird die Kommunikation von Systemen einer Ebene der rechnerintegrierten Fertigung untereinander verstanden. Die dezentralen Steuerungssysteme übernehmen zugleich auch eine Art Filterfunktion, da sie nur die Informationen weitergeben, die in den anderen Rechnersystemen benötigt werden.

Der derzeitige Stand des Fortschreitens der Dezentralisierung ist durch den beginnenden Einsatz sogenannter *intelligenter Feldgeräte* gekennzeichnet. Die in diesen, direkt mit dem technischen

Prozeß in Berührung stehenden, Geräten vorhandene lokale Verarbeitungskapazität wird zur Entlastung der dezentralen Steuerungsgeräte genutzt. Als Begleiterscheinung tritt dabei ein Bedarf an horizontaler Kommunikation zwischen den Feldgeräten, z.B. bei der Ausführung einfacher Regelungsapplikationen ohne Beteiligung der überlagerten Steuerungssysteme, auf. Für die Zukunft ist anzunehmen, daß die Dezentralisierung weiter fortschreitet und auch die heute noch weitgehend ausgesparten Bereiche betreffen wird, in denen die Ausführung der Applikationen - und damit auch der Austausch von Daten zwischen den an der Anwendung beteiligten Systemen - harten Echtzeitanforderungen (vgl. Kap. 2.4) unterliegt.

2.2 Fertigungssysteme als verteilte Systeme

Basierend auf den Ausführungen von *Enslow* [12], *Herrtwich et al.* [13] und *Dietsch* [14] wird der Begriff des *Verteilten Systems* in dieser Arbeit wie folgt gebraucht:

> *Ein **Verteiltes System** ist ein Datenverarbeitungssystem, dessen Daten- oder auch Funktionskomponenten auf mehrere, mittels eines Kommunikationssystems miteinander verbundene Rechner (Knoten) verteilt sind, die eine gemeinsame Gesamtaufgabe in kooperativer Autonomie erfüllen.*

Dabei spielt sowohl der Aspekt der örtlichen als auch der funktionellen Verteilung als Charakteristikum eine Rolle. Die örtliche Verteilung betrifft sowohl die Hardware als auch die Software. Charakteristisch für verteilte Hardware ist das Nicht-Vorhandensein eines gemeinsamen Speichers und eine nicht vernachlässigbare Dauer für den Austausch von Informationen zwischen zwei unterschiedlichen Systemen [15]. Mit der örtlichen Verteilung der Hardware geht die Verteilung der Prozesse und damit auch der Informationsverarbeitung einher. Die Einzelsysteme können dabei durchaus heterogener Natur sein. Gleiches gilt für die benutzten Kommunikationssysteme.

Die Notwendigkeit, in verteilten Systemen auch die Kontrolle über das System zu verteilen, führt zu einer Autonomie der Einzelsysteme. Darin sind insbesondere Funktionen des Selbsttests und der Ausnahmebehandlung, z.B. beim Ausfall des Kommunikationssystems, vorzusehen. Autonomie der Knoten erlaubt damit ein höheres Fehlertoleranzniveau und die Einführung einer kontrollierten Redundanz [16, 17]. Voraussetzung für eine Autonomie der Knoten ist wiederum das Vorhandensein einer lokalen Verarbeitungskapazität, die durch einen Mikroprozessor oder einen Mikrocontroller zusammen mit der benötigten Peripherie, z.B. Speicher und Ein-und Ausgabeschnittstellen, erbracht werden kann. Diese lokale Verarbeitungskapazität erlaubt auch eine selbständige Bearbeitung der Teile der Gesamtaufgabe, die dem Knoten zugeordnet sind [16, 18].

Fertigungssysteme weisen eine inhärente örtliche Ausdehnung auf. Zudem sind in dezentralen Systemen, wie in Kapitel 2.1 dargestellt, auch die Steuerungssysteme mit den ihnen zugeordneten lokalen Aufgaben verteilt. Die einzelnen Rechnersysteme sind dabei in der Regel heterogen und werden z.B. durch Workstations, PCs, SPSen oder intelligente Feldgeräte, z.B. Sensoren oder Aktoren mit lokaler Verarbeitungsleistung und Kommunikationsanschaltung, gebildet [19].

Die gesamte Rechneranlage zur Steuerung der Produktion kann daher als ein *verteiltes System* gemäß der obigen Begriffsbestimmung betrachtet werden. Es existiert eine gemeinsame Aufgabe - die Abwicklung der Fertigung bei Einhaltung technischer und ökonomischer Randbedingungen - und der Bedarf, Informationen zwischen den physisch verteilten Rechensystemen auszutauschen. Dieser Informationsaustausch überspannt in diesem Fall alle Ebenen der rechnerintegrierten Fertigung [1]. Es können aber auch Teilsysteme davon als untergeordnete, weitgehend autonome, verteilte Systeme betrachtet werden. Bild 5 verdeutlicht dies am Beispiel einer Fertigungs-

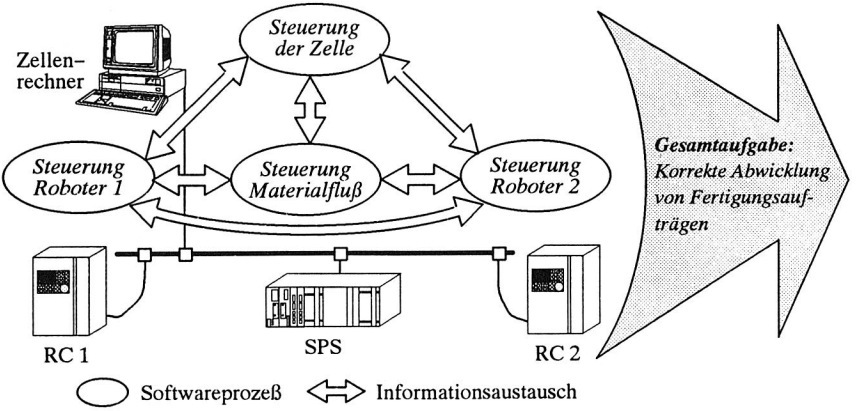


Bild 5: Zellensteuerung als verteiltes System

zelle mit übergeordnetem Zellenrechner, zwei Robotersteuerungen (Robot Control, RC) und einer SPS. Die Steuerungssysteme bearbeiten jeweils eine lokale Aufgabe und kommunizieren dazu mit den anderen Systemen zur Erfüllung der gemeinsamen Aufgabe.

2.3 Kommunikation in der rechnerintegrierten Fertigung

Der im verteilten System einer rechnerintegrierten Fertigung notwendige Austausch von Informationen, der in diesem Kontext als dritter Produktionsfaktor neben Arbeit und Kapital genannt wird [1], erfolgt dort mittels des Einsatzes verschiedenartiger Kommunikationssysteme [20]. Die Zuordnung der unterschiedlichen Kommunikationssysteme zu funktionalen Bereichen eines Unternehmens wird nachfolgend kurz ausgeführt, bevor vertiefend auf prozeßnahe Kommunikationssysteme eingegangen wird.

2.3.1 Hierarchisches Kommunikationsmodell

Entsprechend der hierarchischen Steuerungsstruktur läßt sich auch die rechnerintegrierte Produktion ebenenartig strukturieren [21]. Sowohl bezüglich der Anzahl der Ebenen als auch bezüglich der ihnen zugeordneten Bezeichnungen gibt es dabei in der Literatur unterschiedliche Darstellungen. Ein diesbezüglicher Überblick wird z.B. in [22] gegeben.

Eine adäquate Kommunikationsinfrastruktur wird eingesetzt, um die Kommunikation zwischen den Rechnerapplikationen der verschiedenen Ebenen zu ermöglichen. Dabei ist es aufgrund sowohl technischer als auch wirtschaftlicher Randbedingungen nicht in sinnvoller Weise möglich, ein einziges Kommunikationssystem zur Erfüllung aller Kommunikationsanforderungen einzusetzen. Vielmehr ist es erforderlich, in den einzelnen Ebenen auf die spezifischen Bedürfnisse ausgerichtete Kommunikationssysteme zu verwenden [23]. Die unterschiedlichen Anforderungen der einzelnen Bereiche an das Kommunikationssystem sind in Bild 6 zusammenfassend qualitativ dargestellt.

Schon seit geraumer Zeit werden die Kommunikationsbedürfnisse in den Ebenen oberhalb der Prozeßebene durch die Vernetzung der dort angesiedelten Rechnersysteme befriedigt [24]. Dabei

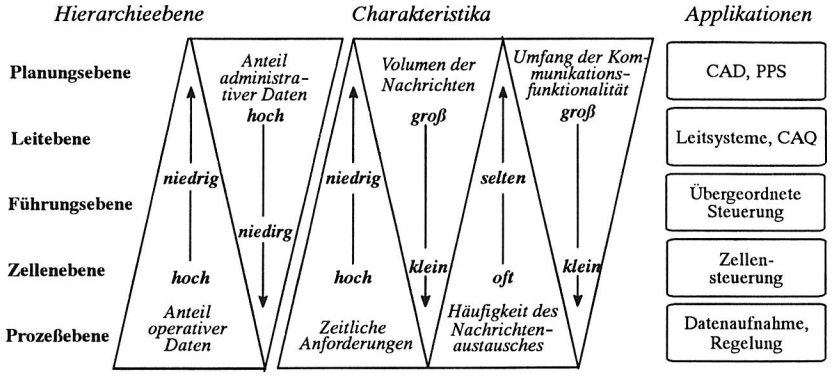


Bild 6: Anforderungen an die Kommunikation in der rechnerintegrierten Fertigung

kommen im allgemeinen derzeit primär herstellerspezifische Kommunikationssysteme, wie z.B. *SNA* oder *DECnet* [25], und Industriestandards wie TCP/IP [26] zur Anwendung. Die Vorteile, die durch offene, standardisierte Kommunikationssysteme geboten werden, führen zu einem zunehmenden Einsatz standardisierter Kommunikationsarchitekturen und begleitend zu einer Ablösung nicht standardisierter Lösungen [27]. Ein derartiger Übergang ist jedoch nur schrittweise im Rahmen einer *Migrationsstrategie* sinnvoll möglich [28, 29, 30].

In der untersten Ebene, der Prozeßebene, finden digitale Kommunikationssysteme eine stark zunehmende Verbreitung. Diese Entwicklung führt zur Ablösung analoger, direkter Verbindungen durch digitale, speziell auf die Kommunikationsbedürfnisse in den unteren Hierarchieebenen der rechnerintegrierten Fertigung abgestimmte, Kommunikationssysteme [24, 31, 32, 33].

2.3.2 Standardisierte Kommunikationsarchitekturen: MAP und TOP

In den höheren Ebenen der rechnerintegrierten Fertigung finden zwei Protokollprofile besondere Beachtung: *MAP* und *TOP*. Das MAP (Manufacturing Automation Protocol)-Profil wurde seit 1985 basierend auf den Anforderungen der Anwender definiert und deckt weitgehend die Kommunikationsbedürfnisse der Leit-, Führungs- und Zellenebene ab. Dabei werden für alle sieben Schichten des ISO-Basisreferenzmodells [8] genormte Protokolle verwendet. Als aufwandsärmere und kostengünstigere Variante existiert zudem die MiniMAP-Architektur, die für die Schichten 3–6 keine explizite Ausprägung von Protokollen vorsieht. Als wichtigste Implementierung der MiniMAP Spezifikation gilt das japanische FAIS [36]. Übergänge zwischen MAP und MiniMAP Netzwerksegmenten können durch Stationen realisiert werden, die gemäß der *Enhanced Performance Architecture* (EPA) aufgebaut sind und damit sowohl einen MAP- als auch einen MiniMAP-Protokollstapel enthalten. Abgestimmt auf die Bedürfnisse der Büroautomatisierung wird MAP durch das TOP (Technical and Office Protocol)-Profil für die Leit- und Planungsebene ergänzt, das ebenfalls vollständig auf standardisierten Kommunikationsprotokollen basiert [20, 34].

2.3.3 Das anwendungsunterstützende Protokoll „MMS"

Ein wesentlicher Grund für die Bedeutung von MAP ist die Definition des zu MAP gehörigen Protokolls MMS (Manufacturing Message Specification) [35]. Dieses Protokoll wurde konsequent

auf die Belange der Steuerung von Fertigungssystemen hin ausgelegt und hat für die Konzeption von anderen Protokollen der Anwendungsschicht im Bereich der rechnerintegrierten Produktion große Bedeutung erlangt. So lassen sich angepaßte Versionen z.B. in den Systemen PROFIBUS, FIP (vgl. zu beiden Kapitel 3.4) und FAIS [36] finden.

Als zentrale Bestandteile des Konzeptes von MMS sind die *Objektbasierung* [37] und die Verwendung des *Client-Server*-Modells [38] anzuführen. Objekte dienen zur Modellierung von realen Fertigungssystemen und deren Ressourcen. Zusätzlich existieren Objektklassen zur Verwaltung der Kommunikation, wie z.B. die Klasse der *Transaction*-Objekte. Die durch die Benutzung von Objekten ermöglichte Abstraktion vom realen Gerät, seiner speziellen Ausprägung und Funktionsweise, erlaubt es dem Benutzer, verschiedenartige Geräte auf die gleiche Art anzusprechen. So kann das *Virtuelle Fertigungsgerät (VMD, Virtual Manufacturing Device)* einen Industrieroboter mit seiner Steuerung ebenso repräsentieren, wie z.B. auch einen Bestückautomaten.

Auf die vorhandenen Objekte kann mittels definierter Dienste zugegriffen werden. Die Dienste sind der Übersichtlichkeit halber in Dienstgruppen zusammengefaßt. Abhängig vom Typ der Objekte können dabei auch neue Objektinstanzen erzeugt und vorhandene Instanzen gelöscht werden. Eine Übersicht über die Funktion der einzelnen Dienste wird z.B. von *Blumann et al.* [39] gegeben.

2.3.4 Feldkommunikationssysteme - Definition und Einordnung

Die Thematik der Kommunikation in den prozeßnahen Bereichen der rechnerintegrierten Fertigung ist ein zentraler Bestandteil dieser Arbeit. Nach einer einführenden Begriffsbestimmung erfolgt in diesem Abschnitt eine allgemeine Einordnung von Feldkommunikationssystemen aus Sicht der Informatik und - aus der Sicht der Automatisierungstechnik - eine Einordnung in die informationstechnische Infrastruktur der rechnerintegrierten Produktion. Dabei wird im folgenden der Terminus „Feldkommunikationssystem" anstelle des Begriffes „Feldbus" benutzt, da letzterer durch die Aufnahme des Begriffes „Bus" eine nicht erstrebenswerte Einschränkung der Systeme auf solche mit bestimmten physikalischen Eigenschaften andeutet, die auch einige der heutigen „Feldbusse", z.B. der INTERBUS-S [40] mit seiner Topologie als physikalischer Ring, nicht erfüllen und die aus Sicht der Anwendung nicht relevant sind.

Lange Zeit waren für die Kommunikation im Prozeßbereich analoge Schnittstellen von großer Bedeutung. Als herausragende Lösungen sind hier die analoge 4/20 mA Strom- und die 0–10 V Spannungsschnittstelle für den Anschluß von Feldgeräten, z.B. Meßwertaufnehmern oder Ventilen, zu nennen. Mit diesen Festlegungen liegen elektrische Normen vor, die eine einfache Austauschbarkeit der Geräte erlauben. Die Anbindung der Feldgeräte geschieht mittels einer signalorientierte Einzelverdrahtung. Die Übertragung analoger Meßwerte erfolgt dabei nach einer Normierung ebenfalls analog in Form einer bestimmten Stromstärke bzw. einer Spannung. Zusätzlich ist dabei noch eine Zuordnung der übertragenen Werte zu der physikalischen Referenzgröße des Meßwertes, z.B. die Zuordnung einer Spannung zu einer Temperatur, erforderlich [31, 41].

Diese Zuordnung entfällt bei der Einführung digitaler Kommunikationssysteme. Neben dem digitalisierten Meßwert, der theoretisch in beliebiger Auflösung übertragen werden kann, sind zusätzliche Aussagen über den Informationsgehalt sowie Statusinformationen über das Gerät übermittelbar. Zudem erlauben Feldkommunikationssysteme einen bidirektionalen Informationsfluß, wohingegen bei herkömmlichen analogen Schnittstellen nur eine unidirektionale Datenübertragung möglich war. Das Vorliegen der Information in digitaler Form erlaubt auch die für den Anwender wichtige Informationsintegration [20].

Als Zwischenschritt auf dem Weg von der analogen Technik zu digitalen Feldkommunikationssystemen ist die SMART-Technologie anzusehen. Hier wird der Prozeßwert analog übertragen, die Konfigurierung und Sonderfunktionen hingegen werden mittels aufmodulierter digitaler Signale angesprochen [42, 43]. Eine Gegenüberstellung der wesentlichen Charakteristika von Feldkommunikationssystemen, der SMART-Technologie und konventionell verdrahteten Systemen ist in Tabelle 1 dargestellt. Bezüglich der SMART-Technologie werden deren analoger (a) und digitaler (b) Bestandteil unterschieden.

Kriterium \ Systemtyp	Konventionell	SMART	Feldkommunikationssysteme
Art der Datenübertragung	analog	a) analog b) digital	digital
Unterstützte Kommunikationsrichtungen pro Leitungspaar	unidirektional	a) unidirektional b) bidirektional	bidirektional
Informationsgehalt der Signale / Nachrichten	niedrig	a) niedrig b) hoch	hoch
Verfügbarkeit des Gesamtsystems	hoch	hoch	a priori: niedrig
Hilfsenergieversorgung	einfach	einfach	begrenzt möglich
Komplexität der Schnittstelle	niedrig	a) niedrig b) mittel	hoch

Tabelle 1: Gegenüberstellung wesentlicher Charakteristika von konventioneller Verdrahtung, SMART- und Feldkommunikationssystemen

Begriffsbildung „Feldkommunikationssystem"

In der Literatur gibt es derzeit noch keine eindeutige Definition des Begriffes „Feldkommunikationssystem" bzw. „Feldbus". In Übereinstimmung mit den inhaltlichen Gemeinsamkeiten dieser Begriffsfestlegungen werden Feldkommunikationssysteme im Kontext dieser Arbeit wie folgt definiert:

Definition 1: Feldkommunikationssystem

Ein Feldkommunikationssystem ist ein digitales Kommunikationssystem, das in verteilten Automatisierungssystemen zur Erfüllung der Kommunikationsanforderungen derjenigen Automatisierungsgeräte dient, die in der untersten Ebene der Hierarchie der rechnerintegrierten Fertigung angesiedelt sind.

Das bedeutet, daß Feldkommunikationssysteme die Kommunikation zwischen Feldgeräten, Sensoren und Aktoren untereinander und mit den überlagerten Steuerungssystemen ermöglichen. Eine Kommunikation zwischen Geräten der Zellenebene ist damit in der Regel ebenfalls möglich.

Damit existiert eine Abgrenzung der Feldkommunikationssysteme von den herkömmlichen analogen Verbindungen, Systemen in SMART-Technologie und von digitalen Kommunikationssystemen für andere Anwendungsbereiche, z.B. Büroumgebungen. Vielfach ist in der Literatur eine weitere Klassifizierung von prozeßnahen, digitalen Kommunikationssystemen zu finden. So wird auch von *Sensor/Aktor-Bussen* oder *Zellbussen* gesprochen. Diese zusätzlichen Unterscheidungen sind aber eher politisch als technisch bedingt und werden hier nicht weiter betrachtet.

Mit PROFIBUS und FIP werden zwei standardisierte Feldkommunikationssysteme, die auf dem Markt größere Bedeutung erlangt haben und in der Automatisierungstechnik allgemein einsetzbar

sind, in Kapitel 3.4 vorgestellt. Daneben gibt es weitere Feldkommunikationssysteme und verwandte Architekturen, die sich von typischen Feldkommunikationssystemen unterscheiden, weil

- sie ursprünglich für andere Zielbereiche konzipiert wurden bzw.
- nur eng abgegrenzte Einsatzfelder in der Automatisierungstechnik haben.

Zu ersteren zählen beispielsweise CAN [44] und ABUS [45], die speziell für den Einsatz in Automobilen konzipiert wurden. Zu letzteren gehören z.B. SERCOS [46] und ASI [47], aber auch der INTERBUS-S [40], der rein für den zyklischen Datenaustausch optimiert wurde. Beim *SERCOS-Interface* handelt es sich hingegen um ein Kommunikationssystem, das nur für die digitale Kommunikation zwischen Steuerungen und Antrieben innerhalb von numerisch gesteuerten Werkzeugmaschinen konzipiert wurde. *ASI* wiederum ist als System allein zur Anbindung einfachster binärer Sensoren und Aktoren anzusehen.

Einordnung von Feldkommunikationssystemen

Der Einsatzbereich von Feldkommunikationssystemen in der Automatisierungstechnik ist bereits Bestandteil der Begriffsfestlegung. Zu den Applikationen derartiger Kommunikationssysteme gehört z.B. der Austausch von Soll- und Istwerten in Lageregelkreisen numerisch gesteuerter Achsen von Handhabungssystemen und die Übertragung von Alarm- und Statusmeldungen. Der Einsatz von Feldkommunikationssystemen unterliegt dabei einer Reihe allgemeiner Anforderungen, die im Überblick in Bild 7 dargestellt sind.

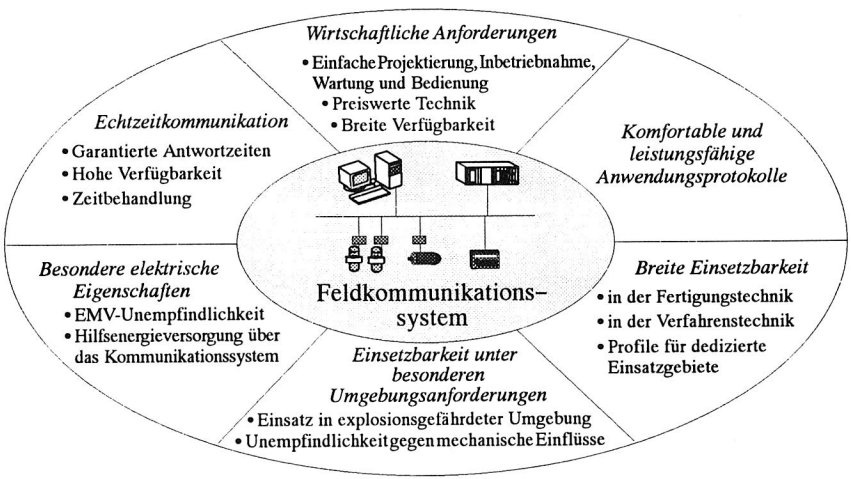


Bild 7: Allgemeine Anforderungen an Feldkommunikationssysteme

Üblicherweise sind größenordnungsmäßig bis zu 100 Endgeräte an ein Feldkommunikationssystem anzuschließen. Die Ausdehnung eines Netzwerkes kann – je nach Einsatzgebiet – wenige Meter bis einige Kilometer betragen. Die anzuschließenden Geräte sind dabei u.U. sehr heterogen: es müssen sowohl intelligente Sensoren und Aktoren als auch leistungsfähige Steuerungssysteme und PCs angeschlossen werden. Daneben besteht aus Sicht der Anwender der Wunsch nach Ver-

fügbarkeit von standardisierten Systemen, die den Aufbau von Anlagen erlauben, die aus interoperablen Komponenten verschiedener Hersteller bestehen [20, 48].

Die innerhalb der Prozeß- und zwischen der Prozeß- und der Zellenebene zu übertragenen Informationseinheiten sind dabei in der Regel nur wenige Byte lang, müssen aber häufig in einem deterministischen zyklischen Zeitraster ausgetauscht werden, um z.B. den Anforderungen von Regelungsapplikation zu genügen (vgl. Bild 6). Die Auffrischungsintervalle für Informationen liegen dabei im Bereich von wenigen Millisekunden bis zu einigen Sekunden [44]. Auf die weiteren Anforderungen bezüglich *Echtzeitkommunikation* wird in Kapitel 3.1 näher eingegangen. Eine detaillierte Beschreibung der *wirtschaftlichen Anforderungen* findet sich z.B. in der *User requirements* Studie des FICIM Projektes [49]. Quantitative Darstellungen der Anforderungen sind, geordnet nach Branchen, in [50] dargestellt.

In der Informatik werden Kommunikationssysteme bezüglich mehrerer Kriterien klassifiziert. Hierzu zählen insbesondere der *Einsatzzweck*, die *geographische Ausdehnung* sowie die *Topologie*. Im Hinblick auf den Einsatzzweck werden Kommunikationssysteme nach *Kauffels* [25] dahingehend eingeordnet, ob sie die Basis eines

- *Datenverbundes,* d.h., der Verbindung von Rechensystemen zur gemeinsamen Nutzung von Datenbeständen,
- *Funktionsverbundes,* z.B. zur Ermöglichung des entfernten Zugriffes auf spezielle Systeme, wie Großrechner oder teure Peripherie,
- *Verfügbarkeitsverbundes,* d.h., der Verbindung mehrerer Rechner zum Zwecke der Erhöhung der Verfügbarkeit des Gesamtsystems,
- *Leistungsverbundes,* d.h., der Verbindung mehrerer Rechner zur Erhöhung des Durchsatzes, oder eines
- *Lastverbundes,* dem Zusammenschluß von Rechnersystemen, in dem die Last dynamisch auf die Systeme aufgeteilt wird, die aktuell den geringsten Auslastungsgrad aufweisen, bilden.

Feldkommunikationssysteme dienen insbesondere dem Datenverbund und dem Verfügbarkeitsverbund. Neuerdings existieren Ansätze, die – ermöglicht durch den Einsatz von Feldkommunikationssystemen – in Richtung eines Lastverbundes zielen [51]. Dabei sind durch die direkten Anbindung der Feldgeräte an den technischen Prozeß und durch die Anforderungen an das Echtzeitverhalten prinzipielle Einschränkungen, z.B. bezüglich einer anvisierten *Prozeßmigration*, also der dynamische Verteilung von Rechenprozessen, gegeben [52].

Die aus der Anwendung kommenden Forderungen nach deterministischer, durchsatzorientierter Kommunikation und preiswerter Technik führen dazu, daß verglichen mit dem ISO-Basisreferenzmodell für die Verbindung offener Systeme [8], nur Protokolle für drei der dort vorgesehenen sieben Schichten explizit ausgeprägt sind. Es handelt sich dabei, wie durch Bild 8 veranschaulicht wird, um die Bitübertragungsschicht, die Sicherungsschicht und die Anwendungsschicht. Die aus den nicht implementierten Schichten benötigte Funktionalität, z.B. zur Kodierung und Dekodierung von Daten, wird den ausgeprägten Protokollen zugeschlagen. Bei einzelnen Systemen, wie z.B. PROFIBUS-DP (vgl. Kapitel 3.4.1), wird auch auf ein Protokoll für die Anwendungsschicht verzichtet. Statt dessen werden Mechanismen der Sicherungsschicht über eine definierte Schnittstelle direkt dem Benutzer zur Verfügung gestellt.

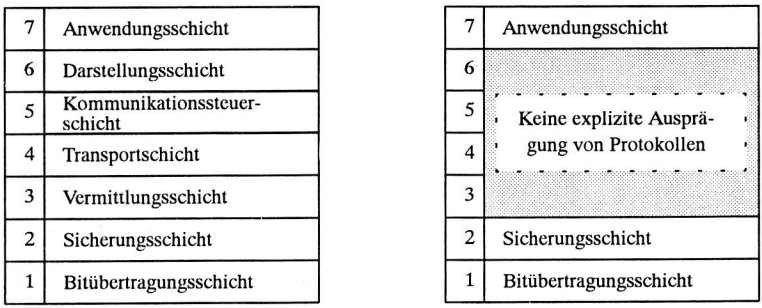


Bild 8: Reduzierte Schichtenarchitektur in Feldkommunikationssystemen

2.4 Echtzeit

Die Verarbeitung von Daten in Echtzeit spielt seit geraumer Zeit insbesondere in der Automatisierungstechnik eine große Rolle. In der jüngeren Vergangenheit sind, was primär auf die Verfügbarkeit preiswerter leistungsfähiger hochintegrierter Schaltkreise zurückzuführen ist, weitere Anwendungsbereiche hinzugekommen. Dazu zählen beispielsweise Steuerungssysteme im Hausgebrauch (z.B. Hausgerätesteuerungen), in Verkehrssystemen (z.B. ABS im KFZ, Bahnleitsysteme, Luftverkehrsüberwachung) aber auch in der Medizintechnik (z.B. Computer-Tomographen) [52, 53]. In diesem Abschnitt werden später benötigte Definitionen für die grundlegenden Begriffe angeführt. Die Begriffe „Echtzeit" und „Realzeit" werden in den nachfolgenden Ausführungen - dem allgemeinen Sprachgebrauch folgend - synonym gebraucht.

2.4.1 Begriffsbildung

Die Forderung nach *Echtzeitverarbeitung* bedeutet, daß bestimmte Aktionen, z.B. Rechenoperationen oder Kommunikationsvorgänge, bis zu einem gegebenen Zeitpunkt korrekt abgeschlossen sein müssen [5]. Damit wird eines der grundlegenden Charakteristika von Echtzeitsystemen im Vergleich zu normalen Rechnersystemen aufgezeigt: In Echtzeitsystemen hängt die Korrektheit eines Ergebnisses nicht nur von der logischen Richtigkeit der zugrundeliegenden Berechnung ab, sondern auch von der Rechtzeitigkeit der Bereitstellung des Ergebnisses. Ein weiteres Charakteristikum ist die explizite Einführung der Dimension *Zeit* [52].

Speziell auf den Realzeitbetrieb von Rechnersystemen ausgerichtet, läßt die in der Norm DIN 44300 [54] gegebene Definition bezüglich der *Realzeitverarbeitung* auch die Übertragung auf andere Anwendungsfelder zu und sei deshalb im folgenden zitiert:

> *„Eine Verarbeitungsart, bei der Programme zur Verarbeitung anfallender Daten ständig ablaufbereit sind, derart, daß die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind."*

Diese Definition wird noch ergänzt um eine Charakterisierung der Zeitpunkte, zu denen Daten anfallen können: *„Die Daten können je nach Anwendungsfall nach einer zufälligen Verteilung oder zu vorbestimmten Zeitpunkten anfallen."*

Wesentlich ist in diesem Zusammenhang auch der Aspekt der *Zuverlässigkeit* des Systems. Echtzeitanforderungen können nur durch ein funktionsfähiges System erbracht werden. Mechanismen zur Erhöhung der Zuverlässigkeit sind sowohl in Hard- als auch in Software vorzusehen. Dazu zählen etwa redundante Hardwarekomponenten und Mehrheitsentscheidungen [5, 19].

Fertigungssysteme sind technische Anlagen, bei denen ein u.U. auf hohem Energieniveau ablaufender technischer Prozeß kontrolliert und geführt werden muß. Die dazu eingesetzten Rechnersysteme, die häufig verteilt angeordnet sind, müssen die Programme derart abarbeiten, daß sie mit dem Prozeß Schritt halten [52, 55]. Damit wird gefordert, daß, wie in Bild 9 dargestellt, sowohl die einzelnen verteilten Rechnersysteme, das in ihnen benutzte Betriebssystem und auch das sie miteinander verbindende Kommunikationssystem echtzeitfähig sein müssen. Die im weiteren Verlauf der Arbeit benutzten Begriffe zu dieser Thematik werden wie folgt definiert:

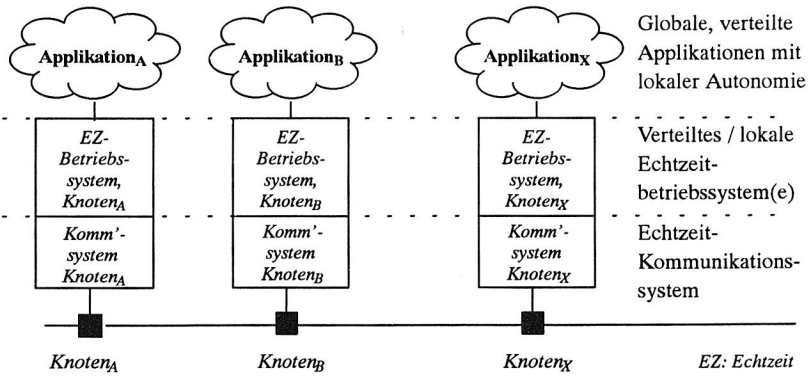


Bild 9: Systemarchitektur eines verteilten Echtzeitsystems

Definition 2: Echtzeitdienst

*Ein **Echtzeitdienst** ist ein Dienst, der innerhalb einer bestimmten, durch die Umgebungsanforderungen vorgegebenen Zeitspanne ausgeführt wird [19]. Diese Zeitspanne erstreckt sich über das Interval [R_S, D_S], wobei R_S (Release) den frühesten erlaubten und D_S (Deadline) den spätesten noch gültigen Zeitpunkt für die Beendigung des Dienstes angibt.*

In vielen Fällen wird R_S nicht explizit angegeben, sondern ist durch den Zeitpunkt des Dienstaufrufes implizit gegeben. Einige Autoren (z.B. *Powell* [19]) führen zudem noch eine Zielzeit T_S (Targettime) ein, die den Zeitpunkt markiert, zu dem der Nutzen der Dienstausführung am größten, bzw. die damit verbunden Kosten am niedrigsten sind.

Definition 3: Echtzeitsystem

*Ein **Echtzeitsystem** ist ein System, das mindestens einen Echtzeitdienst erbringt.*

Definition 3 besagt insbesondere auch, daß in einem Echtzeitsystem nicht jeder angebotene Dienst ein Echtzeitdienst sein muß. In den meisten Anwendungssituationen unterliegt nur eine Teilmenge der geforderten Dienste Echtzeitanforderungen [15, 19]. Definition 3 läßt sich für verteilte Systeme wie folgt erweitern:

Definition 4: Verteiltes Echtzeitsystem
*Ein **verteiltes Echtzeitsystem** ist ein verteiltes System, das mindestens einen Echtzeitdienst erbringt [19].*

Im Einsatzbereich von Feldkommunikationssystemen werden Informationen über den Zustand des technischen Prozesses in Variablen modelliert. Diese Prozeßvariablen sind wie folgt definiert:

Definition 5: Prozeßvariable
Eine Prozeßvariable ist die Instanz einer Variablen, die einen Prozeßwert repräsentiert.

In verteilten Systemen kann, bedingt durch die Übertragungszeit der Information zwischen der erzeugenden und der verbrauchenden Applikation, die Information bei ihrem Eintreffen beim Verbraucher schon veraltet sein. Aus diesem Grunde sind Vorkehrungen notwendig, die es erlauben, die zeitliche Gültigkeit von ausgewählten Prozeßvariablen in Echtzeit-Prozeßvariablen zu modellieren [56].

Definition 6: Echtzeit-Prozeßvariable
*Eine **Echtzeit-Prozeßvariable** ist ein Prozeßvariable, die einen Prozeßwert und die zu diesem gehörige Zeitinformationen zusammenfaßt. Diese Zeitinformationen beinhalten den Erstellungszeitpunkt, den Gültigkeitszeitraum des Prozeßwertes und Datenkonsistenzattribute.*

Die Begriffe „Echtzeitvariable" und „Echtzeit-Prozeßvariable" werden im folgenden synonym gebraucht.

Je nach Auswirkung bei Überschreitung der Zielzeit D_S eines Echtzeitdienstes wird zwischen *harten* und *weichen Zeitschranken* unterschieden [19, 55, 57].

Definition 7: Harte Zeitschranke
*Eine **harte Zeitschranke** ist ein Zielzeitpunkt D_S, bei dessen Verfehlung die zugeordnete Aktion keinen Nutzwert mehr hat oder sogar Schaden anrichten kann.*

Definition 8: Weiche Zeitschranke
*Eine **weiche Zeitschranke** ist ein Zielzeitpunkt D_S, bei dessen Verfehlung die zugeordnete Aktion je nach Verspätung weniger Wert hat bzw. höhere Kosten verursacht.*

2.4.2 Prinzipielle Beschränkungen

Die in der DIN 44300 beschriebenen Charakteristik des Auftretens von Daten impliziert, daß das Echtzeitdatenverarbeitungssystem auch zu zufälligen Zeitpunkten auftretende Anforderungen in Echtzeit verarbeiten kann. Diese Forderung ist realiter unerfüllbar, da jeder Bearbeitungsvorgang einer Anforderung beliebig oft und damit auch beliebig lange durch Anforderungen mit höherer Dringlichkeit unterbrochen werden kann [55]. Auch die Erfüllung von Echtzeitanforderungen zweier, sehr kurz nacheinander eintreffender höchstpriorer Anforderungen kann nicht garantiert werden, wenn die Bearbeitungszeit mindestens einer der beiden Anforderungen größer oder gleich der maximal erlaubten Reaktionszeit der anderen ist.

Eine weitere Einschränkung ergibt sich aus der prinzipiell vorhandenen, zufallsbedingten Wahrscheinlichkeit des Ausfalls von Komponenten dergestalt, daß auch die vorgesehenen Sicherheitsmechanismen keine Verfügbarkeit der Anlage mehr gewährleisten können. Damit ist auch eine Garantie für die Einhaltung von Zeitschranken nicht möglich [55].

2.4.3 Fertigungssysteme als verteilte Echtzeitsysteme

Fertigungssysteme lassen sich subsumierend als verteilte Echtzeitsysteme beschreiben. Die physische Verteilung ergibt sich aus der Struktur der technischen Anlage mit einer angepaßten Modularisierung und Dezentralisierung der Steuerungsaufgaben. Das notwendige Schritthalten der Steuerungs- und Regelungsfunktionen mit dem technischen Prozeß induziert die Echtzeitanforderungen. Der bestehende Bedarf, Informationen zwischen den verteilt angeordneten Komponenten des Systemen auszutauschen muß durch entsprechend ausgelegte Kommunikationssysteme befriedigt werden.

2.5 Einsatz von Multimedia-Technologie in den prozeßnahen Bereichen

In den nachfolgenden Abschnitten werden die Grundlagen der Multimedia-Technologie und der in den prozeßnahen Bereichen der rechnerintegrierten Produktion bestehende Bedarf für multimediale Applikationen vorgestellt. Darauf aufbauend werden Feldkommunikationssysteme als Instrument auch zur Übertragung multimedialer Informationen motiviert. Als Ziel wird dabei ein *digital integriertes* System angestrebt. In dieser Ausbaustufe eines Multimedia-Systems finden sowohl die Datenübertragung als auch die Übertragung von Kontrollinformationen vollständig digital über ein gemeinsames Kommunikationssystem, in diesem Fall ein Feldkommunikationssystem, statt (vgl. Bild 10). Die durch die Integration zusätzlich entstehenden Probleme dürfen dabei nicht dergestalt gelöst werden, daß das System für seinen ursprünglichen Einsatzzweck nicht mehr brauchbar ist [58, 59].

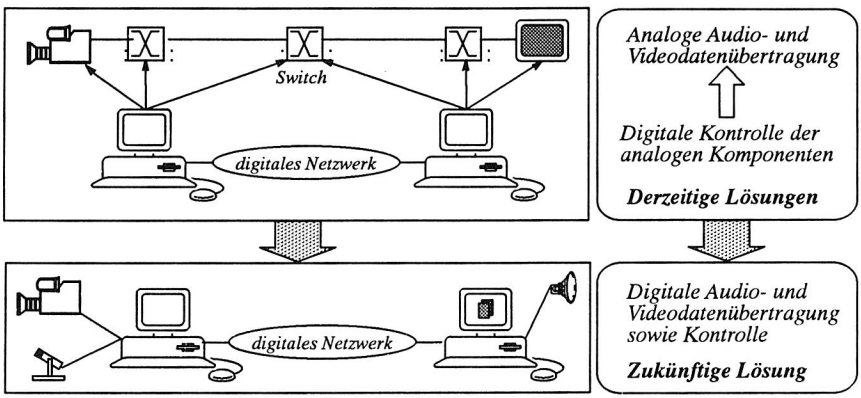


Bild 10: Entwicklung zu digital integrierten Multimedia-Systemen (nach [60])

2.5.1 Begriffsbildung

Als Basis für die weitere Benutzung des Begriffs eines Multimedia-Systems dient die von *Steinmetz* [60] eingeführte Definition derartiger Systeme:

> *„Ein Multimedia-System ist durch die rechnergesteuerte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die in mindestens einem kontinuierlichen (zeitabhängigen) und einem diskreten (zeitunabhängigen) Medium kodiert sind."*

Diese Definition ist auch auf Einzelplatzsysteme anwendbar. Besondere Bedeutung erlangen Multimedia-Systeme und -Applikationen jedoch erst in einer verteilten Anwendungsumgebung.

Zu den in der Definition angesprochenen kontinuierlichen Medien gehören z.B. Sprache und Bewegtbilder, wohingegen z.B. Einzelbilder zu den diskreten Medien gezählt werden. Durch obige Definition werden Multimedia-Systeme im engeren Sinne charakterisiert. Nach *Steinmetz* [58] kann aber auch dann von einem Multimedia-System (im weiteren Sinn) gesprochen werden, wenn kein kontinuierliches Medium vorliegt, aber Medien verschiedener Art, z.B. Einzelbilder und Text, verarbeitet werden. Im Kern gleichlautende Definitionen werden auch von anderen Autoren, z.B. *Rakow et al.* [61], angeführt.

2.5.2 Multimedia im betrieblichen Umfeld

Die rechnerintegrierte Fertigung ist, wie verschiedene andere Umgebungen, z.B. Krankenhäuser oder Vertriebs- und Auskunftssysteme, auch, ein Anwendungsfeld für Multimedia-Applikationen. Die wesentlichen Bereiche in diesem Anwendungsfeld sind dabei bisher die

- rechnergestützte Gruppenarbeit (*Computer Supported Co-operative Work, CSCW*) und die
- rechnergestützte Schulung und Ausbildung (*Computer Based Training, CBT*).

Die rechnergestützte Gruppenarbeit hat zum Ziel, eine adäquate Arbeitsunterstützung für die Zusammenarbeit räumlich getrennter Arbeitsgruppen zu schaffen. Bestandteile derartiger Systeme sind üblicherweise Telekonferenzsysteme, gemeinsam nutzbare Notizblöcke und die gemeinsame Bearbeitung von Dokumenten, Zeichnungen etc. [62]. In der rechnergestützten Schulung und Ausbildung wird durch den Einsatz von Multimedia-Systemen die Anschaulichkeit des gelehrten Stoffes verbessert [63, 64, 65].

Über diese etablierten Anwendungsbereiche hinaus besteht in zunehmendem Maße der Bedarf, unterschiedliche Informationen in verschiedenen Darstellungsformen aus den prozeßnahen Bereichen der rechnerintegrierten Fertigung zu gewinnen und zu verarbeiten. Dabei ist die Durchgängigkeit des Informations- und Kommunikationssystems durch alle Ebenen der rechnerintegrierten Fertigung auch für derartige Multimedia-Daten anzustreben [66]. Auf diesen Aspekt wird hier vertiefend eingegangen.

2.5.3 Multimedia im prozeßnahen Bereich

In den prozeßnahen Bereichen der rechnerintegrierten Fertigung werden die aus dem Prozeß gewonnenen Informationen dazu genutzt, neue Sollwerte für Aktoren zu generieren und dem Bediener Informationen über den Prozeßzustand zur Verfügung zu stellen [67]. Dazu wird heute in der Regel der Wert entweder an sich in textueller Form oder seine grafische Repräsentation verwendet. So kann die Temperatur, z.B. in einem Lötofen, mittels analoger oder digitaler Anzeigegeräte, aber auch in Form eines Balkens in einer zum Prozeßwert proportionalen Länge auf einem lokalen oder entfernten Sichtgerät dargestellt werden [68].

Jede Erweiterung des Angebotes führt dabei zu einer qualitativen Verbesserung und damit zu einer Erleichterung der menschlichen Tätigkeit [58]. Moderne Konzepte für die Gestaltung von Bedienoberflächen bieten dazu entsprechende Strukturierungsmechanismen für die verschiedenen Informationen an [69]. So ist es z.B. sinnvoll, Informationen nur dann zur Verfügung zu stellen, wenn der Benutzer sie explizit anfordert („on demand") oder sich eine Ausnahmesituation anbahnt bzw. schon eingetreten ist, in der zusätzliche Informationen für den Benutzer hilfreich sein kön-

nen. Das Vorliegen aller Daten in digitaler Form erlaubt zudem eine einfachere Weiterverarbeitung, z.B. zur Dokumentation, Archivierung oder Analyse (vgl. Bild 11).

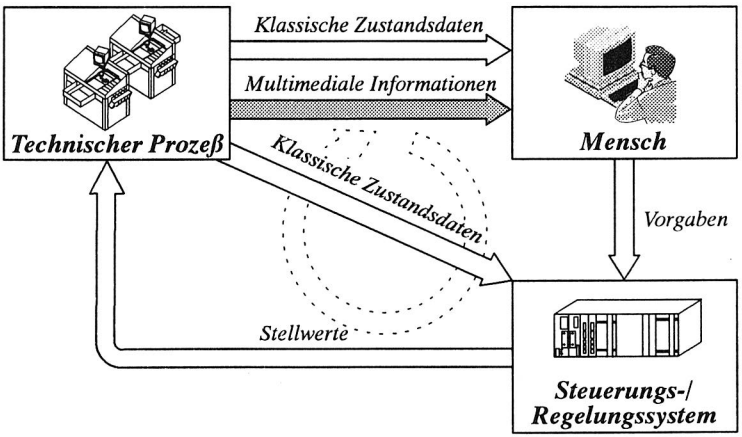


Bild 11: Konventionelle und multimediale Datenströme in der Prozeßebene

Im Regelfall ist heutzutage an Fertigungsleitständen aufgrund der gegebenen räumlichen Trennung von Leitstand und technischem Prozeß jedoch nur eine Teilmenge der für den menschlichen Bediener perzeptiblen Informationen verfügbar. In elektronischen Leitständen können nach *Kurbel* [70] die herkömmlicherweise eingesetzten Darstellungsformen von Informationen, das sind Grafiken und Schriftzeichen, durch Einzelbilder, Bewegtbilder und Audio-Informationen ergänzt werden, um die Abwicklung von Aufgaben in der Fertigungssteuerung zu unterstützen. Im einzelnen werden dabei aufgeführt:

- *Bilder* und *Fotografien* als ergänzende Informationsträger zu Endprodukten, Bauteilen, Maschinen und anderen Objekten,
- *Bewegtbilder* in Form von gespeicherten und später abgerufenen Bildsequenzen sowie zur on-line Überwachung des technischen Prozesses und zur Diagnose und
- *Audio-Informationen* als Ergänzung zu visueller Information und in Form von gesprochenen Nachrichten (sog. *Voice-Mails*) für die Benutzerkommunikation.

Bewegtbild- und Audio-Informationen können dabei in der, durch eine räumliche Trennung charakterisierten Bediener-Prozeß-Schnittstelle, als „Auge" bzw. „Ohr" zum Prozeß dienen [71]. Die Systeme, insbesondere die zur Aufnahme visueller Informationen, können dabei auch über spezielle Eigenschaften verfügen und z.B. als Röntgensystem, Infrarotkamera oder Ultraschall-Bildsystem ausgeführt sein.

Multimediale Informationssysteme bilden auch einen wesentlichen Bestandteil von innovativen Ferndiagnosesystemen, wie z.B. von dem System, das von *Adam et al.* [72] vorgestellt wird. Die dabei benötigten Informationen werden aus der Fertigung über lokale und Weitverkehrsnetze zum Zielsystem übertragen, dort textuell, grafisch, auditiv oder bildhaft dargestellt und weiterverarbei-

tet, wobei das System allerdings weder vollständig digital integriert ist, noch eine Schnittstelle zu Feldkommunikationssystemen aufweist.

Zusammenfassend lassen sich für den prozeßnahen Bereich der rechnerintegrierten Fertigung folgende Anforderungen an die Unterstützung der Verarbeitung multipler Medien ableiten, die anschließend anhand von Anwendungsfällen noch weiter motiviert werden:

- Unterstützung der Aufnahme und Übertragung von Einzelbildern
- Unterstützung der Bewegtbilddatenübertragung aus dem Prozeß in Echtzeit
- Ermöglichung der Übertragung von Audiodaten in Echtzeit

Einzelbildübertragung

Einzelbilder können von verschiedenartigen Anwendungen genutzt werden. Dazu zählen insbesondere:

- Überwachungsapplikationen, bei denen nur zu bestimmten Zeitpunkten Daten aufgenommen werden müssen, z.B. Vollständigkeitsüberprüfung nach Montagevorgängen.
- Applikationen im Rahmen der Qualitätssicherung, bei denen die aufgenommenen Bilder dazu genutzt werden, Aussagen – z.B. in Form von Maßen und Toleranzen – über die Güte der produzierten Teile zu machen [73].

Eine weitere Detaillierung der Anwendungsfälle in typische Aufgabenstellungen wird z.B. von *Föhr* [74] vorgenommen. Hier soll jedoch nur ein typisches Beispiel weiter ausgeführt werden:

Zu den Überwachungsapplikationen gehört z.B. die Aufnahme der Referenzmarken einer zu bestückenden Leiterplatte vor Beginn des eigentlichen Bestückvorgangs mit dem Ziel, eine mögliche Fehllage oder einen Verzug, die mit mechanischen Zentriervorrichtungen nicht mehr kompensiert werden können, festzustellen. Aus den mittels einer CCD-Kamera aufgenommenen Bildern werden dann Korrekturwerte für das Bestückprogramm errechnet (vgl. Bild 12). Die Kamera läßt

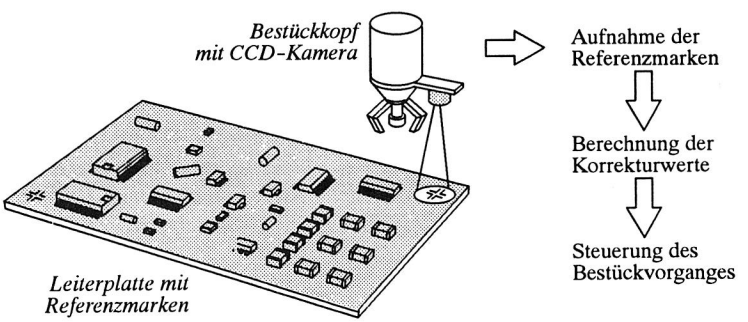


Bild 12: Einzelbildaufnahme zur Berechnung der Korrekturwerte beim Bestücken

sich auch für andere Zwecke, z.B. die Überprüfung der Vollständigkeit der Bestückung, die Lageüberprüfung plazierter Bauteile oder als optisches Identifikationssystem zur Aufnahme eines Strichkodes, einsetzen.

Bewegtbildübertragung

Als wichtigste Applikation für die Bewegtbildübertragung gilt die Visualisierung des technischen Prozesses in Echtzeit. Ziel derartiger Applikationen ist es, dem Bediener trotz der räumlichen Trennung ein ausreichendes Maß an Informationen an der Bediener-Prozeß-Schnittstelle zur Verfügung zu stellen. Dabei muß die Bewegtbildinformation nicht ständig angezeigt werden, sondern kann auch ereignisgesteuert eingeblendet werden. In einigen Anwendungen wird gefordert, daß auf die Informationen zurückgegriffen werden kann, die in einem kurzen Zeitraum vor Eintritt eines Ereignisses aufgenommen wurden [75].

Zinser [71] stellt folgende, auf der Verfügbarkeit von Bewegtbildinformationen basierende, Anwendungen vor:

- Erfassung von Zustandsänderungen des technischen Prozesses, die meßtechnisch sonst nur schwierig aufzunehmen sind (z.B. ausströmender Dampf)
- Prozeßbedienung über im Bewegtbild sichtbare Komponenten
- Überlagerung der angezeigten Bewegtbilder mit Prozeßschemata oder Meßwerten

Übertragung von Audiodaten

Auditive Informationen lassen sich ebenfalls verschiedenartig nutzen:

- Bediener-Bediener Kommunikation zur Verbesserung des Informationsaustausches trotz räumlicher Trennung; diese Funktionalität kann durch sogenannte *Voice-Mailboxen* ergänzt werden.
- Aufnahme von gesprochenen Kommentaren zu anderen, z.B. bildhaften, Informationen
- Überwachung des Prozesses
- Qualitätssicherung

Für die Bediener-Bediener Kommunikation und die Aufnahme gesprochener Kommentare erweist sich die Beschränkung auf den Frequenzbereich bis 3400 Hz (Telefonqualität) i. allg. als ausreichend. Die Weiterverarbeitung des gesprochenen Wortes ist dabei im Rahmen der Spracherkennung zur automatisierten Erstellung von Schrift oder Steuerung von Aktionen möglich [76, 77].

Bei der Prozeßüberwachung ist das gesamte applikationsspezifisch relevante Frequenzspektrum aufzunehmen und wiederzugeben. Die Auswertung von akustischen Signalen findet beispielsweise in der akustischen Montageüberwachung Anwendung. So verursachen z.B. Fügevorgänge typische Reib-, Rast- oder Schnappgeräusche, die durch Überwachungssysteme aufgenommen werden können. Mit Hilfe dieser Daten können unter Verwendung von Verfahren der Mustererkennung auch komplexere Aussagen über die aufgetretene Störung getroffen werden [78]. Für die Qualitätssicherung ist die Analyse des aufgenommenen Geräusches oder Resonanzverhaltens eines Werkstückes eine wertvolle Hilfe [79].

2.6 Multimedia-Basistechnologie

Die Integration der Verarbeitung zusätzlicher Medien in verteilten Systemen birgt zwei zu behandelnde Problemstellungen in sich: Zum einen die aufgrund des großen Datenvolumens notwendige Datenkompression und zum anderen angepaßte Kommunikationssysteme für Multimedia-Anwendungen. Vielfach werden die Begriffe „Kompression" und „Kodierung" in der Literatur fälschlicherweise synonym gebraucht. Im folgenden wird der umfassendere Begriff „Kodierung" verwendet. Die vorgestellten Kodierungsverfahren beinhalten in der Regel auch eine Kompression von Daten.

2.6.1 Datenkodierung

Die bei der Übertragung und Weiterverarbeitung von Multimedia-Informationen anfallenden großen Datenvolumina erfordern eine Kodierung mit Komprimierung der Daten. Durch die Komprimierung verringert sich zum einen die benötigte Übertragungsbandbreite und zum anderen sinkt der Bedarf an Speicherplatz bei der Verarbeitung und der Archivierung. Von den Kodierungsverfahren wird gefordert, daß bei einer geringen Komplexität des Algorithmus die Qualität der Information nach deren Kodierung und Dekodierung noch ausreichend gut ist. Insbesondere in Dialoganwendungen, z.B. Wechselsprechen, wird zudem die Einhaltung einer oberen Zeitschranke für die Kodierung und die Dekodierung gefordert [60]. Die meisten Multimedia-Systeme basieren auf einer Kombination mehrerer der nachfolgend kurz vorgestellten Kodierungsverfahren.

Verfahren der *Entropiekodierung* beruhen auf der Ausnutzung der unterschiedlichen Häufigkeit von Symbolen, z.B. des Auftretens bestimmter Buchstaben in einem Text. Häufig auftretende Symbole werden mit besonders kurzen Codewörtern verschlüsselt, wodurch insgesamt eine Reduktion des Datenvolumens eintritt. Die ursprüngliche Information ist durch die Dekodierung vollständig wiederherstellbar („verlustfreie Kodierung"). Zu den wichtigsten Verfahren der Entropiekodierung zählen die Huffmann-Kodierung und die arithmetische Kodierung [80, 81].

Verfahren der *Quellenkodierung* verwenden die Semantik der behandelten Informationen. Für unterschiedliche Medientypen sind deshalb verschiedene Verfahren unterschiedlich gut geeignet. Die wichtigsten Verfahren werden im Zusammenhang mit der Betrachtung der Kodierung der einzelnen Medientypen nachfolgend kurz vorgestellt. Allgemein betrachtet beruhen die Verfahren der Quellenkodierung auf zwei Prinzipien: der Elimination redundanter und der Unterdrückung irrelevanter Informationen, d.h. von Informationen die entweder aus den bereits vorliegenden Informationen gewonnen werden können oder deren Verlust für den menschlichen Betrachter aufgrund der physioakustischen bzw. -visuellen Charakteristika der Perzeptionsorgane nicht wahrgenommen wird.

Allgemeines zur Kodierung visueller Informationen

Bei der Kodierung von Einzel- oder Bewegtbildern werden bei der Quellenkodierung Verfahren der *Transformation* und der *Prädiktion* in Verbindung mit einer *Quantisierung* genutzt. Die Transformation überführt den Ursprungsraum in den Frequenzbereich, in dem eine weitere Verarbeitung aufgrund der Charakteristika der Daten einfacher möglich ist. Für die Bildverarbeitung hat aufgrund ihrer Eigenschaften die *Diskrete Cosinus Transformation* (DCT) besondere Bedeutung erlangt. Diese überführt eine Gruppe von *NxN* Informationswerten, die zu Bildpunkten gehören, in *NxN* Ortsfrequenzkomponenten.

Die Prädiktion ist ein Verfahren zur Kodierung von Differenzen örtlich benachbarter oder zeitlich aufeinanderfolgender Informationswerte. Anstelle des Absolutwertes wird nur die Differenz zu den benachbarten Werten kodiert. Dabei treten bei den typischerweise flächenhaften Bildinformationen viele kleine und gleiche Werte auf, die sich günstig weiterverarbeiten lassen.

Sowohl die Transformation als auch die Prädiktion gehören, abgesehen von den durch Rechenungenauigkeiten auftretenden Fehlern, zu den verlustfreien Kodierungsverfahren. Die diesen Verfahren häufig nachgeordnete Quantisierung ist hingegen ein verlustbehaftetes Verfahren, daß auf der ganzzahlig gerundeten Division der Informationswerte durch die Elemente einer anwendungsspezifischen Quantisierungsmatrix basiert.

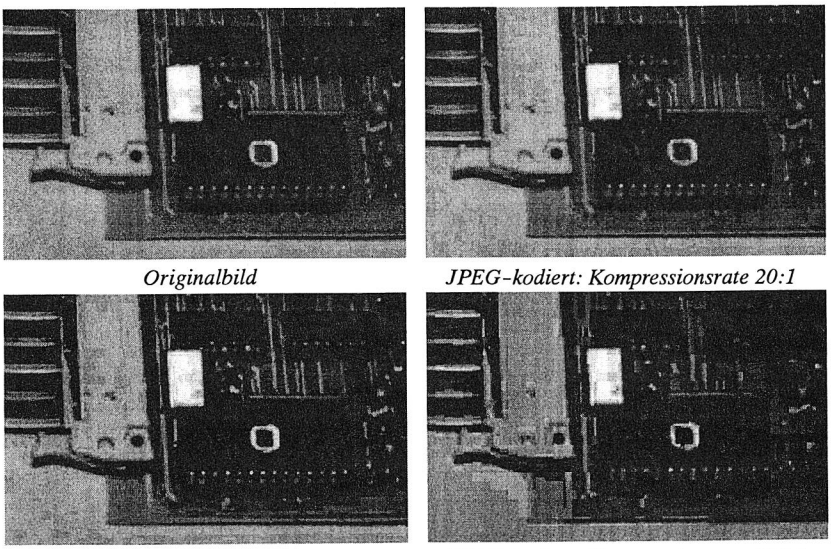
Kodierung von Einzelbildern

Der derzeit maßgebende Standard für die Kodierung von farbigen oder grauskalierten Einzelbildern ist das seit 1992 von der ISO und der ITU-T (International Telecommunications Union-Telecommunication Standardization Sector) genormte JPEG (Joint Photographic Experts Group)-Verfahren [82]. Die Größe sowie das Verhältnis von Höhe zu Breite des zu kodierenden Bildes sind beliebig. Durch Parameterisierbarkeit des Verfahrens kann eine individuelle Abwägung zwischen Qualität des reproduzierten Bildes, Dauer der Kompression und Kompressionsfaktor vorgenommen werden. Grundsätzlich werden zwei verlustbehaftete Modi, ein verlustfreier und ein Modus mit Kodierung des Bildes in verschiedenen Auflösungen unterschieden. Jeder JPEG-Codec (Coder/Decoder) muß den einfachsten Modus, die verlustbehaftete, sequentielle, DCT-basierte Kodierung, unterstützen. Auch der zweite verlustbehaftete Modus von JPEG basiert auf der DCT mit nachfolgender Quantisierung der Transformationskoeffizienten. Durch unterschiedliche Quantisierung können dabei unterschiedlich hohe Kompressionsfaktoren erreicht werden. Für den verlustfreien Modus wird ein Prädiktionsverfahren eingesetzt [83].

Die mit dem JPEG-Algorithmus erreichbaren Kompressionraten liegen bei 10:1 bis 20:1 bei verlustbehafteter Kodierung ohne sichtbare Verluste, 30:1 bis 50:1 mit leichten Qualitätseinbußen und einer maximalen Kompression von 100:1 [84]. Die Auswirkung der Kodierung eines Einzelbildes mit verschiedenen Kompressionsfaktoren ist beispielhaft in Bild 13 dargestellt, wobei dort bei hohen Kompressionsfaktoren auch die blockbasierte Arbeitsweise deutlich wird.

Bewegtbildkodierung

Für die Kodierung von Bewegtbildern existieren mehrere unterschiedliche Verfahren. Zu den derzeit wichtigsten zählen die Empfehlung H.261 der ITU-T und MPEG (Motion Pictures Experts



Originalbild | *JPEG-kodiert: Kompressionsrate 20:1*

JPEG-kodiert: Kompressionsrate 40:1 | *JPEG-kodiert: Kompressionsrate 100:1*

Bild 13: Auswirkungen der verlustbehafteten JPEG-Kodierung

Group). Daneben gibt es weitere Verfahren, wie z.B. AVI (Audio Video Interleaved) der Firma Microsoft, DVI (Digital Video Interactive) der Firma Intel oder Quicktime der Firma Apple [81], auf die hier jedoch aufgrund von deren Status als proprietäre Lösungen nicht weiter eingegangen wird.

❍ Empfehlung H.261 der ITU-T

Im Rahmen der ITU-T Empfehlung H.320 [85] für die *audiovisuelle Kommunikation über $p \times 64$-kbit/s-ISDN-Leitungen* beschreibt die Empfehlung H.261 [86] die Methoden zur Kodierung und Dekodierung der Video-Komponente mit p im Bereich zwischen 1 und 30.

Aus einem Eingangssignal mit einer festgelegten Bildwechselfrequenz von $\approx$29,97 Bildern pro Sekunde werden kodierte Bildsequenzen im *Common Intermediate Format* (CIF) mit 288 Zeilen zu je 352 Pixeln oder im QCIF (Quarter CIF) mit 144 Zeilen zu je 176 Pixeln erzeugt. Jeder Dekoder muß mindestens in der Lage sein, in QCIF kodierte Bildsequenzen verarbeiten zu können.

Die Empfehlung H.261 beschreibt ein hybrides Kodierungsverfahren, das auf Prädiktion, DC-Transformation und Quantisierung beruht. Die Prädiktion basiert auf der Differenzbildung der Werte der aktuellen Bildpunkte zu denen des vorhergehenden Bildes („INTER-Modus"). Dabei werden jeweils sog. *Makroblöcke* betrachtet, die einen Ausschnitt des Bildes zusammenfassen. Spätestens alle 132 Bilder wird jeder Makroblock ohne Prädiktion übertragen um die durch die Prädiktion und die inversen Transformationen zustandegekommenen Fehler auszugleichen („INTRA-Modus"). Optional kann die Kodierung durch eine Bewegungskompensation oder eine Filterung des Bildes mittels eines optischen Tiefpasses ergänzt werden [87]. Der durch den Koder erzeugt Bitstrom enthält Informationen zur Vorwärts-Fehlerkorrektur. Die Nutzung dieser Informationen durch den Dekoder ist optional.

❍ MPEG

MPEG (Moving Pictures Expert Group) ist ein durch die ISO/IEC 1993 im Standard IS 11172 festgelegtes Verfahren primär zur Speicherung von Bewegtbildern und den dazugehörigen Audiodaten. Als maximale Datenrate werden 1,856 Mbps vorgeschrieben. Für die kodierten Bewegtbildinformationen sind dabei bis zu 1,5 Mbps reserviert. Die verbleibende Bandbreite ist für die Übertragung von kodierten Audiodaten vorgesehen. Der Standard beinhaltet eine Festlegung von erlaubten Bildformaten sowie eine Auswahl von möglichen Bildwechselfrequenzen. Typischerweise werden Bilder mit 288 Zeilen zu je 352 Pixeln und einer Bildwechselrate von 24-30 Hz kodiert [80, 88].

Die berücksichtigten Anforderungen nach wahlfreiem Zugriff und effizienter Kodierung führte zu der Definition von drei wesentlichen Bildtypen: In *Intra-Coded-Pictures* („I") werden Einzelbilder in Echtzeit kodiert. *Predictive-Coded-Pictures* („P") werden mit Hilfe der Informationen der vorangegangenen Bilder erzeugt. Die höchste Kompressionsrate wird mit *Bidirectionally-Predictive-Coded-Pictures* („B") erreicht, die basierend auf den Informationen vorangegangener und nachfolgender Bilder kodiert werden. Eine sinnvolle Anordnung von Bildtypen in nach MPEG kodierten Bewegtbilddatenströmen ist in Bild 14 dargestellt.

Eine Weiterentwicklung dieses auch als MPEG-1 bezeichneten Verfahrens betrifft die Anpassung des Verfahrens an Fernsehnormen mit höheren Auflösungen und mit Zeilenverschränkung, z.B. an den Standard CCIR-601 mit 576 (PAL) bzw. 480 (NTSC) Zeilen zu je 704 Pixeln. Der Datenstrom bei diesem mit MPEG-2 bezeichneten Verfahren hat ein Volumen von 4-8 Mbps [84]. Eine

Skalierung des MPEG-Verfahrens läßt sich am günstigsten durch die Variation der Bildrate erreichen [87].

Reihenfolge bei der Kodierung:

Bildart: I P B B P B B I B B...
Bildnr.: 1 4 2 3 7 5 6 10 8 9...

Reihenfolge bei der Darstellung:

Bildart: I B B P B B P B B I...
Bildnr.: 1 2 3 4 5 6 7 8 9 10...

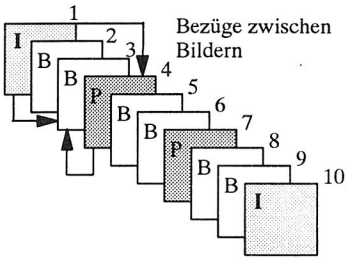


Bild 14: Anordnung von I-, P- und B-Bildern bei MPEG-Kompression

❍ M-JPEG

Bei M-JPEG handelt es sich um eine sequentielle Übertragung von Bildern, die nach dem JPEG-Verfahren kodiert sind. Damit liegt eine von Bild zu Bild hoch redundante, reine Intraframe-Kodierung vor. Die Skalierbarkeit dieses Verfahrens betrifft primär die Bildrate [87].

Kodierung von Audiodaten

Audiodaten, z.B. Sprache, Musik oder Geräusche, gehören zu den kontinuierlichen Medien. Im Regelfall wird das Signal abgetastet und quantisiert (*Pulse-Code-Modulation,* PCM). Für Sprache reicht eine Abtastung mit 8 kHz und eine 8-Bit Quantisierung, so daß der resultierende Datenstrom ein Volumen von 64 kbps hat. Anschließend kann ein Kodierungsverfahren angewendet werden. Das Verfahren der *Differentiellen PCM* (DPCM) basiert auf der Kodierung der Differenz eines aktuellen Wertes zu dessen Vorgänger. Bei vergleichbarer Qualität sinkt die Datenrate von 64 kbps für den mittels PCM erzeugten Datenstrom auf 56 kbps bei Anwendung der DPCM. Die *Adaptive* DPCM (ADPCM) kodiert nicht nur die Differenz, sondern auch einen Faktor für die Änderung der Differenz und erlaubt bei vergleichbarer Qualität eine weitere Absenkung der Datenrate auf 32 kbps. Ausprägungen dieser Verfahren sind durch das ITU-T in den Empfehlungen G.711 (PCM, Abtastrate: 8 kHz) und G.722 (ADPCM, Abtastrate: 16 kHz) standardisiert [60, 84].

Ein weiteres wichtiges Kompressionsverfahren für Audiodaten ist das im Standard ISO/IEC IS 11172-3 definierte MPEG-Audio, die Spezifikation der Audiodatenkodierung des MPEG-Standards. Dieser Standard beinhaltet drei Qualitätsstufen, die sich hinsichtlich Aufwand und Kodierungsgüte unterscheiden. Alle Verfahren bieten alternativ die Abtastraten 32 KHz, 44,1 kHz und 48 kHz an. Der komprimierte Datenstrom hat ein anvisiertes Volumen von 192, 128 bzw. 64 kbps pro Audiodatenkanal [60, 80].

2.6.2 Kommunikationsanforderungen in verteilten Multimedia-Systemen

Bezüglich der Übertragung von Multimedia-Daten muß strikt zwischen den diskreten und den kontinuierlichen Medien unterschieden werden. Bei diskreten Daten sind keine besonderen kommunikationstechnischen Voraussetzungen zu berücksichtigen. Sie können in der Regel bei Verfügbarkeit der gesamten Information zeitlich entkoppelt von deren Erstellung zwischen den Systemen ausgetauscht werden [89]. Bei der Übertragung von Daten kontinuierlicher Medien müssen dagegen bestimmte, durch den Medientyp gegebene Randbedingungen eingehalten werden. Diese betreffen

- den Durchsatz, der durch die Datenrate des Datenstromes definiert wird,
- die maximale Verzögerung zwischen der Datenquelle und der Datensenke,
- die maximale Varianz beim Eintreffen der Daten in der Datensenke (*Jitter*) und
- die Paketverlustrate [90].

So muß die Übertragung von kontinuierlichen Medien, z.B. Bewegtbild- und Audiodaten, isochron zwischen Quelle und Senke erfolgen. Damit ist auch der Jitter in der Übertragungszeit beschränkt. Einzelne Fristverletzungen, fehlerhafte oder fehlende Informationseinheiten fallen jedoch aufgrund der physioakustischen und -visuellen Eigenschaften der menschlichen Perzeptionsorgane kaum ins Gewicht.

Die benötigte Datenrate für die Übertragung kontinuierlicher Medien ist direkt abhängig von der Periodizität und der Länge der vom Koder erzeugten, zu übertragenden Informationseinheiten. Ist der von dem Koder erzeugte Datenstrom *streng periodisch* [60] und *streng gleichmäßig*, d.h. variiert er weder in der zeitlichen Abfolge der Erzeugung von Informationseinheiten noch in deren Größe, so können die Bandbreitenanforderungen an das Kommunikationssystem exakt angegeben werden. Ist er nur *schwach periodisch* oder variiert die Größe der einzelnen Informationseinheiten mit einer gegebenen Periodizität (*schwache Gleichmäßigkeit*), so kann dies auf Kosten der Reservierung von Ressourcen und zusätzlicher Verzögerung der Dateneinheiten oder Verschwendung von Kommunikationsbandbreite durch lokale Zwischenspeicherung bzw. die ständige Reservierung der maximal benötigten Kommunikationsbandbreite kompensiert werden. Ist er hingegen *aperiodisch* oder *ungleichmäßig*, so können höchstens statistische Aussagen über die zu einem bestimmten Zeitpunkt benötigte Bandbreite gemacht werden, auf deren Basis die Parameter des Kommunikationssystems gesetzt werden können.

Die wichtigsten Kodierungsverfahren für kontinuierliche Medien (H.261, [A]DPCM, MPEG) erzeugen zumindest schwach gleichmäßige und schwach periodische Datenströme, so daß sich eine Obergrenze für die benötigte Kommunikationsbandbreite angeben läßt. Die Übertragung derartiger Datenströme sollte dann *isochron*, d.h. mit festen zeitlichen Abständen zwischen den einzelnen Informationseinheiten, unter Echzeitbedingungen mit möglichst geringer Ende-zu-Ende Verzögerung erfolgen. Die weiteren Anforderungen an das Kommunikationssystem werden in Kapitel 3.3 vorgestellt. Die quantifizierten medienbezogenen Kommunikationsanforderungen in verteilten Multimedia-Systemen sind in Tabelle 2 zusammenfassend dargestellt.

	Audio (Sprache)	**Video**		**Daten**
		unkomprimiert	komprimiert	
Datenrate	16–64 kbps	10–100 Mbps	64 kbps–100 Mbps	~0,1–10 Mbps
Ende zu Ende Verzögerung	~ 250 ms	~ 250 ms	~ 250 ms	~ 1000 ms
Tolerabler Jitter	~ 10 ms	~ 10 ms	~ 1 ms	~ 1 ms
Paketverlustrate	10^{-2}	10^{-2}	10^{-11}	10^{-11}

Tabelle 2: Medienbezogene Kommunikationsanforderungen (nach Steinmetz [60])

Bild 15 stellt ergänzend die geforderten Datenübertragungsraten von mit verschiedenen Kodierungsverfahren behandelten Bewegtbilddatenströmen den typischen Datenübertragungsraten in Feldkommunikationssystemen gegenüber.

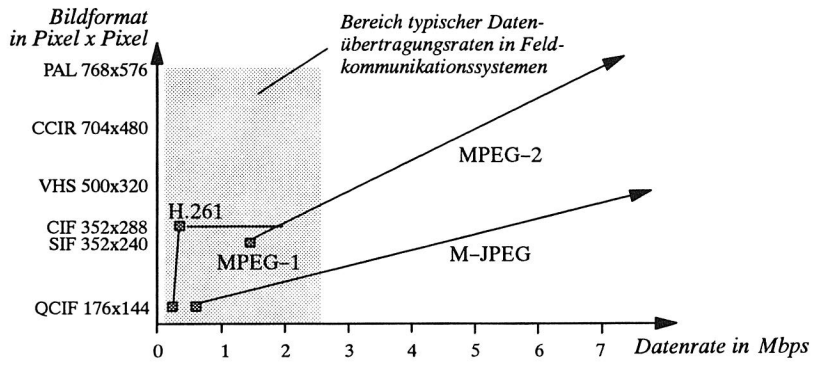


Bild 15: Bewegtbildkodierung: Erforderliche Datenrate in Abhängigkeit von Kodierungsverfahren und Bildformat (nach [84])

2.7 Integration in Feldkommunikationssysteme

In den prozeßnahen Bereichen der rechnerintegrierten Fertigung werden Feldkommunikationssysteme zur Erfüllung der Kommunikationsanforderungen eingesetzt. Um ergänzende Netzwerkinstallationen vermeiden zu können ist es das Ziel, auch das aus dem Einsatz von digital integrierten Multimedia-Anwendungen entstehende zusätzliche Kommunikationsaufkommen durch Feldkommunikationssysteme abzudecken (vgl. Bild 16) [59].

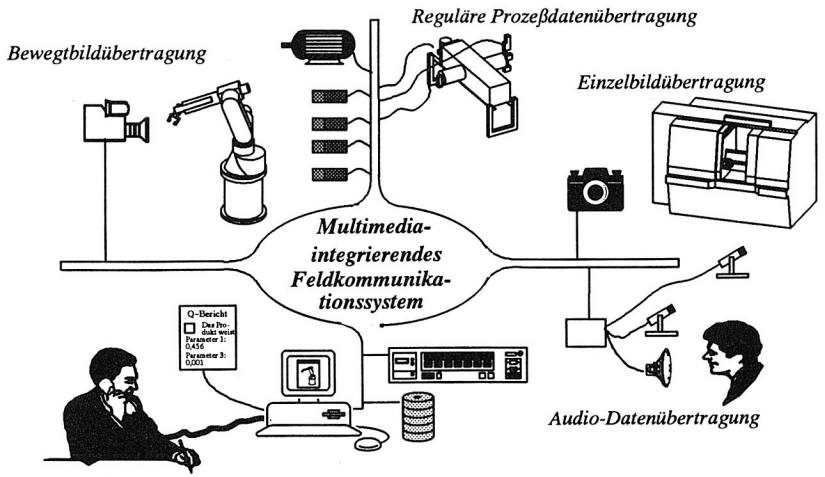


Bild 16: Anwendungsfelder für Multimedia-Dienstintegration

2.7.1 Adäquate Kodierungsverfahren beim Einsatz von Feldkommunikationssystemen

Die Charakteristika der Feldkommunikationssysteme, z.B. die begrenzte Bandbreite und die Nutzung für die Übertragung auch von Prozeßdaten, stellen neben den Anforderungen der Anwendung einen weiteren restringierenden Faktor bei der Auswahl eines Kodierungsverfahrens dar. Ohne weitere Betrachtung der Anforderungen der Applikation werden nachfolgend Kodierungsverfahren für die verschiedene Medientypen vorgestellt, die sich bei der Verwendung von Feldkommunikationssystemen anbieten.

Die Übertragung von kodierten Bewegtbildern stellt vom in Echtzeit zu übertragenden Datenvolumen her im Vergleich zu der Übertragung von Audiodaten bzw. Einzelbildern die größten Anforderungen. Für heutige Feldkommunikationssysteme ist das H.261-Verfahren das am besten geeignete Kodierungsprinzip, da es bereits mit einer sehr niedrigen Datenrate von minimal 64 kbps die Übertragung von Bewegtbildern erlaubt, sich durch eine hohe Skalierbarkeit auszeichnet und seine Implementierungen einen streng periodischen und streng gleichmäßigen Datenstrom erzeugen. Die dafür notwendige Skalierung des Koders wird durch das Verfahren selbst vorgenommen. Die Verfügbarkeit einer höheren Bandbreite erlaubt die Erhöhung der Bildwechselfrequenz und die Verwendung des CIF-Formats anstelle des QCIF. Interessant ist zudem die Option, mit vergleichsweise niedrigem Aufwand eine Anbindung an öffentliche Telekommunikationsnetze – insbesondere ISDN – durchführen zu können. Damit wird der Aufbau von Fernüberwachungs- und -bedienungssystemen vereinfacht. Für zukünftige Feldkommunikationssysteme mit höheren verfügbaren Bandbreiten stellen MPEG und M-JPEG Alternativen zu H.261 dar. Durch die Verwendung dieser Verfahren können insbesondere die Restriktionen von H.261 bezüglich der verfügbaren Bildformate überwunden werden.

Als Verfahren für die Kodierung von Einzelbildern ist JPEG prädestiniert. Die hohen, ohne sichtbaren Qualitätsverlust erzielbaren Komprimierungsfaktoren kommen der beschränkten Bandbreite heutiger Feldkommunikationssysteme entgegen. Die Skalierbarkeit des Kodierungsalgorithmus sowie die Freizügigkeit bzgl. des Formates des Ursprungsbildes erlauben die Befriedigung von Anforderungen verschiedenartiger Applikationen. Für die Kodierung von Audiodaten eignen sich die dargestellten, auf PCM basierenden Kodierungsverfahren DPCM und ADPCM, die sich durch einen streng periodischen und streng gleichmäßigen Datenstrom auszeichnen.

2.7.2 Ansätze aus dem Umfeld der Multimedia-Dienstintegration

Ausgewählte Arbeiten, die sich mit der Unterstützung von verteilten Multimedia-Applikationen auseinandersetzen, werden nachfolgend kurz vorgestellt. Im Gegensatz zu dem in dieser Arbeit verfolgten Ansatz der Integration von Unterstützungsfunktionalität für Multimedia-Applikation in Kommunikationssysteme mit vergleichsweise geringer Übertragungsbandbreite werden darin primär Hochgeschwindigkeitsnetze wie z.B. ATM (Asynchronous Transfer Mode [91]) betrachtet. Auch wenn eine einfache Übertragung der Ansätze auf Feldkommunikationssysteme aufgrund deren spezieller Charakteristik nicht in Betracht kommt, so zeichnen sie sich doch dadurch aus, daß auch bei ihnen bereits während der Konzeptionsphase des Systems die das Kommunikationssystem betreffenden Komponenten auf die Behandlung kontinuierlicher Multimedia-Datenströme ausgerichtet wurden. Alle drei vorgestellten Ansätze beinhalten Aspekte, die bei der vorliegenden Arbeit inspirierend eingeflossen sind.

❍ AudioFile

Bei *AudioFile: A Network-Transparent System for Distributed Audio Applications* handelt es sich um ein experimentelles System aus dem *Cambridge Research Laboratory* der Firma DEC. Audio-

File stellt eine abstrakte Benutzerschnittstelle für Audio-Gerätschaften dar. Ein evtl. zwischen dem aufnehmenden und dem wiedergebenden System vorhandenes Netzwerk ist für die darauf aufsetzenden Applikationen transparent.

Die wesentlichen Komponenten von AudioFile sind

- das *Protokoll* zwischen dem AudioFile-Server und dem AudioFile-Client,
- die *Funktionalität* und die *Programmierschnittstelle* für Clients und
- der *Server*, der es erlaubt, von den realen Systemen zu abstrahieren

sowie einer Reihe von beispielhaften Client-Applikationen. Die Client-Server Architektur von AudioFile basiert auf dem Konzept, wie es im X-Window System benutzt wird. Im Regelfall werden die Server-Applikationen erst auf Anforderung mindestens eines Clients hin aktiv. Eine Ausnahme besteht durch die Möglichkeit, Ereignismeldungen asynchron vom Server an den Client zu senden. Den Clients stehen eine Reihe von Funktionen u.a. zur Steuerung der Verbindung und der Geräte, zum Setzen und Modifizieren von Audio-Kontexten und zur Fehlerbehandlung zur Verfügung. Dazu gehört auch die Funktionalität zur Anbindung des lokalen Audio-Gerätes an das Telefonsystem. Der Server besteht aus einem geräteunabhängigen und einem geräteabhängigen Teil, die über eine gemeinsame Schnittstelle miteinander kommunizieren [92].

AudioFile ist insbesondere aufgrund des verfolgten Gedankens der Geräteunabhängigkeit interessant. Die Transparenz bezüglich des Kommunikationssystems wird im Bereich der Integration von Multimedia-Diensten in Feldkommunikationssysteme nicht benötigt und der aus der Komplexität des Ansatzes erwachsende Implementierungsaufwand verbietet zudem den Einsatz im Bereich von Feldkommunikationssystemen.

❍ Tenet

Die Tenet-Arbeitsgruppe der Universität von Berkeley und des *International Computer Science Institute* betrachtet Multimedia-Applikationen als eine besondere Klasse von Echtzeit-Applikationen, deren Kommunikationsanforderungen durch ein echtzeitfähiges, paketorientiertes Kommunikationssystem, wie es im Rahmen des Tenet-Projektes entwickelt wurde, erfüllt werden können.

Im Tenet-System wird dazu ein Reservierungsverfahren angewendet, das auf Anforderung eines Clients hin eine Ende-zu-Ende Verbindung zu einem Server aufbaut. Die Reservierung der benötigten Ressourcen erfolgt beim Verbindungsaufbau. Client und Server können dabei in unterschiedlichen Netzwerken angesiedelt sein. Die Wahl eines festen Weges erfolgt beim Verbindungsaufbau. Stehen nicht ausreichend viele Ressourcen zur Verfügung, so wird eine entsprechende Fehlermeldung an den Client übergeben. Im Falle eines erfolgreichen Verbindungsaufbaus werden die Anwendungsdaten über einen sog. *Realtime-channel* transportiert. Dieser ist als eine Ende-zu-Ende Simplex-Verbindung definiert, auf der die ausgehandelten Übertragungsleistungen garantiert werden können.

Die für das Tenet-System vorgeschlagene Protokollarchitektur beinhaltet fünf Protokolle (vgl. Bild 17), die auf vorhandene Protokolle der Sicherungsschicht, wie z.B. FDDI, aufsetzen. Auf Ebene der Vermittlungsschicht kommt das *RealTime Internet Protocol* (RTIP) zum Einsatz. Darauf setzen zwei Protokolle der Transportschicht auf, die dem Aspekt des unterschiedlichen Charakters von Echtzeit-Nachrichten, wie sie oftmals in Steuerungsanwendungen auftreten, und Echtzeit-Datenströmen von Multimedia-Applikationen Rechnung tragen. Dies sind das *Realtime Message Transport Protocol* (RMTP) und das *Continous Media Transport Protocol* (CMTP). Der Aufbau von *Realtime-channels* wird durch das *Realtime Channel Administration Protocol*

(RCAP) bewerkstelligt. Primär für die Kontrolle der Datenaustausches und die Reaktion auf Fehlerzustände ist das *Realtime Control Message Protocol* (RTCMP) zuständig.

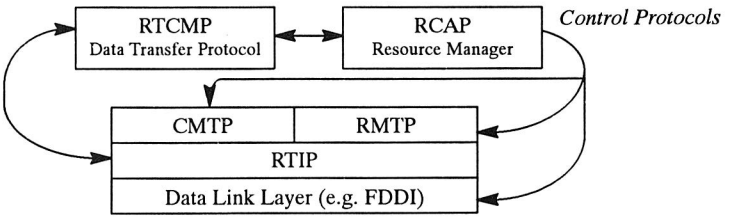


Bild 17: Protokollstapel des Tenet-Systems (nach [93])

An der Dienstschnittstelle werden der aufsetzenden Applikation Parameter zur Spezifikation der Anforderungen zur Verfügung gestellt. Diese beinhalten die minimale und die maximale Paketzwischenankunftszeit, die maximale Paketgröße, die maximal tolerierte Verzögerungszeit nebst der tolerierten Wahrscheinlichkeit eines Überschreitens der Verzögerungszeit, die erlaubte Wahrscheinlichkeit eines Pufferüberlaufs sowie den maximalen Jitter [93].

Das Tenet-System ist bereits prototypisch implementiert worden. Interessant sind die Ansätze, die geforderte Dienstgüte an der Client-Schnittstelle applikationsunabhängig beschreiben zu können sowie die Bevorratung von speziellen Mechanismen zur Übertragung von kontinuierlichen Datenströmen. Als Mängel dieses Ansatzes werden die fehlende Unterstützung von Gruppenkommunikation für Datenströme und Echtzeit-Nachrichten und der hohe Aufwand für die Reservierung von Ressourcen angesehen.

❍ VuNet

Das VuNet wurde am MIT (Massachuchetts Institute of Technology) als *Netzwerkarchitektur für verteilte Multimedia-Systeme* konzipiert. Als Charakteristika dieses Systems sind anzuführen, daß neben den regulären Arbeitsplatzrechnern sog. *Multimedia-Endgeräte* (Multimedia Devices) direkt an das Netzwerk angeschlossen werden können. Seitens der Kommunikation wird ein ATM-System verwendet. Es wird nicht zwischen den verschiedenen Typen von zu übertragenden Daten unterschieden (Datentransparenz). Die Klassifikation des Datentyps (z.B. Teil eines Bewegtbilddatenstromes, Teil eines Dateitransfers, ...) obliegt den Anwendungen. Das Konzept von VuNet sieht vor, daß Multimedia-Daten in einer Multitasking-Umgebung nur zu bestimmten Zeitpunkten bearbeitet werden können. Das resultierende Kommunikationsaufkommen ist dadurch nicht kontinuierlich sondern im allgemeinen sporadisch. Die Notwendigkeit, die verfügbaren Ressourcen in einer derartigen Laufzeitumgebung zu teilen, führt auch zu dem Ansatz der Skalierbarkeit von Bewegtbilddatenquellen und der Möglichkeit, diese entfernt zu kontrollieren. Im VuNet-Projekt wurden des weiteren angepaßte Endgeräte entwickelt. Dabei handelt es sich zum einen um spezielle Kommunikationsgeräte (insbesondere sog. *Switches*) und zum anderen um dedizierte Multimedia-Endgeräte [94].

Die Konzepte der Multimedia-Endgeräte und der entfernten Kontrolle der Datenquellen von kontinuierlichen Datenströmen sind auch für Feldkommunikationssysteme relevant. Die angestrebte Datentransparenz ist für Feldkommunikationssysteme aufgrund der dort in der Regel vorhande-

nen Mechanismen zur Echtzeitkommunikation ebenfalls von Interesse. Die Unterscheidung verschiedener Datentypen erst auf Ebene der Anwendung ist jedoch für den Feldkommunikationsbereich nicht adäquat. Diese hat hier – insbesondere aus Gründen der z.T. nur geringen Verarbeitungsleistung der benutzten Stationen – auf einer der unteren Protokollebenen stattzufinden.

2.8 Zusammenfassung

Basierend auf den sich abzeichnenden Entwicklungen im Bereich der Steuerungstechnik wurden in diesem Kapitel Feldkommunikationssysteme zur Übertragung

- von Informationen, die zur Steuerung dezentraler, technischer Prozesse mit Echtzeitanforderungen notwendig sind als auch
- von multimedialen Daten, die in der Regel für den Bediener bestimmt sind (Bewegtbilder, Standbilder und auditive Daten), aber auch maschinell weiterverarbeitet werden können, motiviert.

Bezüglich beider Aspekte wurden die wesentlichen Grundlagen dargestellt. Diese beinhalten allgemein die Charakterisierung von Feldkommunikationssystemen und – bezüglich der Übertragung von Steuerungsinformationen – die Festlegung einer Menge benötigter Begriffe zur Beschreibung von Echtzeit-Kommunikationssystemen. Im Hinblick auf die Integration der Unterstützung von multimedialen Applikationen wurden die Basisbegriffe der Multimedia-Technologie und die bei der Übertragung über Kommunikationssysteme zu berücksichtigenden Aspekte vorgestellt.

Die besonderen Restriktionen, die bei der Übertragung von Informationen für Multimedia-Applikationen in Feldkommunikationssystemen zu berücksichtigen sind, betreffen – wie ausgeführt wurde – auch die einzusetzenden Kodierungsverfahren. Eine Auswahl relevanter Kodierungsverfahren wurde skizziert und es wurde die besondere Eignung von H.261, JPEG und PCM-basierter Verfahren für die Übertragung von Bewegt- bzw. Standbildern und Audio-Informationen motiviert. Ergänzend wurden drei Ansätze aus dem thematischen Umfeld der Integration von Multimedia-Datenverarbeitung in verteilten Applikationen betrachtet.

3 Anforderungsanalyse und Bewertung aktueller Feldkommunikationssysteme

Dieses Kapitel dient der Vorstellung qualitativer Anforderungen, die sich aufgrund der dargestellten Entwicklungen zukünftig an Feldkommunikationssysteme stellen, sowie der Bewertung ausgewählter, verfügbarer Feldkommunikationssysteme bezüglich der bereits durch sie gewährleisteten Erfüllung dieser Anforderungen. Dazu wird folgende Unterteilung bezüglich der Anforderungen vorgenommen:

- Übertragung von Prozeßdaten in Echtzeit (EZ)
- Unterstützung verteilter Systeme (VS) in den prozeßnahen Bereichen
- Integration der Unterstützung für Multimedia-Applikationen (MM)

3.1 Übertragung von Prozeßdaten in Echtzeit

Die nachfolgend dargestellten Anforderungen bezüglich der Erfüllung von Echtzeitanforderungen sind eine auf die Bedürfnisse von Feldkommunikationssystemen ausgerichtete Zusammenstellung von Kriterien. Sie ist in einer ersten Form bereits als Teil des Anforderungskatalogs für das OLCHFA-Feldkommunikationssystem (vgl. Kap. 4.1) erarbeitet und für dessen Konzeption verwendet worden [95].

ANFORDERUNG ***EZ-1: Rechtzeitigkeit***

Kommunikationsdienste mit Echtzeitanforderung müssen spätestens bis zu der ihnen vorgegebenen Zielzeit ausgeführt worden sein [96]. Die Einhaltung von garantierten Zielzeiten wird sowohl für periodisch anfallende Anforderungen als auch für aperiodische Kommunikationsaufträge gefordert [15].

Der Bedarf nach aperiodisch auszuführenden Echtzeitdiensten besteht im Anwendungsbereich von Feldkommunikationssystemen insbesondere beim Senden von Nachrichten über den Eintritt von Ereignissen im technischen Prozeß („Alarme"). Echtzeitdienste in periodischer Ausführungsform werden in Regelkreisen benötigt, bei denen die Sollwerte für die Aktoren und die durch Sensoren aufgenommenen Istwerte über das Kommunikationssystem von bzw. zum Automatisierungsprogramm, das die Regelung vornimmt, übertragen werden. Die meisten mathematischen Regelmodelle erfordern eine Abtastung des Istwertes zu äquidistanten Zeitpunkten [97].

Die bei der Kommunikation auftretende Verzögerung, die im Regelkreis auch als *Totzeit* bezeichnet wird und in dessen Modell durch ein *Totzeitglied* τ nachgebildet wird (vgl. Bild 18), ist deshalb zum einen gering zu halten und zum anderen ist ihre Schwankung (Jitter), die sich negativ auf die Stabilität des Regelkreises auswirkt, zu minimieren.

ANFORDERUNG ***EZ-2: Berücksichtigung von Echtzeit- und Nicht-Echtzeit Kommunikationsanforderungen***

In fast allen Einsatzfällen für Feldkommunikationssysteme existieren Kommunikationsaufträge mit und solche ohne Echtzeitanforderungen. Für die Aufträge ohne Echtzeitanforderungen wird ebenfalls eine adäquate Unterstützung gefordert. Insbesondere ist der bei der Echtzeitkommunikation zusätzlich anfallende Aufwand zu vermeiden.

ANFORDERUNG ***EZ-3: Vorhersehbarkeit***

Das Verhalten des Kommunikationssystems auf die Einlastung von Echtzeitanforderungen muß vorhersehbar sein. Dies gilt sowohl für den Normalbetrieb als auch für den Betrieb unter Überlast [96].

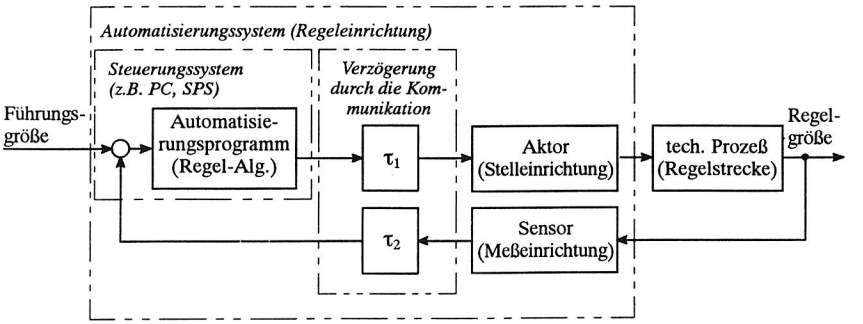


Bild 18: Verzögerung der Meß- und Stellgrößen (nach [33])

ANFORDERUNG ***EZ-4: Zuverlässigkeit***

Da Systeme mit hohen Ausfallwahrscheinlichkeiten nicht geeignet sind, den Anforderungen nach Rechtzeitigkeit und Vorhersehbarkeit gerecht zu werden, ist die Forderung nach Zuverlässigkeit unabdingbar [96]. Hier sind Maßnahmen der Fehlertoleranz und der Fehlererkennung gefordert. In Feldkommunikationssystemen gehören dazu Redundanzmechanismen für Komponenten, die für die Funktionalität des Gesamtsystems von zentraler Bedeutung sind. Zudem sollen Informationen über den festgestellten Ausfall von Komponenten des Systems für den Benutzer zugänglich sein.

ANFORDERUNG ***EZ-5: Verarbeitung von Echtzeit-Prozeßvariablen***

Die Verfügbarkeit von Informationen über die zeitliche Gültigkeit von Informationen gewinnt insbesondere unter dem Aspekt der Verteiltheit eines Systems an Bedeutung [56]. Deshalb wird eine Unterstützung der Verarbeitung von Echtzeit-Prozeßvariablen (vgl. Kap. 2.4.1, Definition 6) gefordert. Insbesondere ist die zeitliche Gültigkeit der Daten anhand der zugehörigen Zeitinformation zu überprüfen, bevor sie an den Benutzer weitergereicht oder von diesem übernommen werden.

Als notwendig wird dazu ein *Zeitstempel* zur Anzeige des Erzeugungszeitpunktes der Daten und ein *Gültigkeitszeitraum* erachtet. Der Beginn der Gültigkeit eines Datenwertes fällt dabei nicht zwingend mit dem Zeitpunkt seiner Erzeugung zusammen. Vielmehr besteht die Möglichkeit, daß sie erst zu einem beliebigen, späteren Zeitpunkt beginnt, wodurch die Möglichkeit geschaffen wird, daß in Abhängigkeit von diesem Zeitpunkt eine Synchronisation von Aktionen im verteilten System stattfindet.

Daneben wird gefordert, daß die Bereitstellung folgender Informationen über die Konsistenz von Daten durch das Kommunikationssystem unterstützt wird [95, 98]:

- *Rechtzeitigkeit in der Verteilung (Local timeliness)*
  Dieses Datenkonsistenzattribut wird beim Konsumenten des Datenwertes gesetzt und zeigt an, ob eine Variable innerhalb der vorgegebenen Zeitspanne über das Netzwerk aktualisiert wurde oder nicht.
- *Rechtzeitigkeit in der Erzeugung (Remote timeliness)*
  Mit Hilfe dieses, auf Seiten des Erzeugers des Datums gesetzten Datenkonsistenzattributes, wird angezeigt, ob das bezogene Datum innerhalb der vorgegebenen Zeitspanne durch die Applikation erzeugt wurde oder nicht.

- *Gültigkeit beim Erzeuger (Remote validity)*
  Dieses Datenkonsistenzattribut erlaubt es dem Erzeuger, den konsumierenden Stationen mitzuteilen, daß der Datenwert aufgrund eines inkonsistenten Zustandes in der Erzeugerstation nicht gültig ist. Hierbei geht es nicht um die zeitliche Gültigkeit des Datenwertes, sondern um andere Einflußfaktoren, z.B. Meßbereichsüberschreitungen, Ausnahmebehandlungen des Betriebssystems etc.

Des weiteren stellt der Zeitstempel eine Referenzmarke für den Gültigkeitszeitraum dar und ermöglicht auch die Evaluierung der Werte der Attribute *„Local Timeliness"* und *„Remote Timeliness"*. Die Granularität des Zeitstempels und die Gültigkeitsdauer einzelner Echtzeitvariablen sind gemäß der Anforderungen im Bereich der Feldkommunikation festzulegen (vgl. auch Anforderung 14).

ANFORDERUNG EZ-6: Unterstützung einer verteilten Uhr

Feldkommunikationssysteme, eingesetzt als integraler Bestandteil von verteilten Echtzeitsystemen zur Steuerung technischer Komplexe, ermöglichen den Austausch von Informationen zwischen den einzelnen, verteilten Komponenten. Als Basis einer Synchronisation der ablaufenden verteilten Applikationen gewinnt eine in allen Stationen verfügbare, synchronisierte Uhrzeit große Bedeutung [99].

Dabei ist ein logischer Uhrzeitmechanismus, wie er z.B. von *Lamport* [100] vorgestellt wird, nicht ausreichend, da er zwar abhängig vom Fortschritt des Prozesses Zeitstempel mit monoton steigenden Werten liefert, diese aber keinen Bezug zur realen Zeit herstellen lassen [101]. Damit kann eine logische Uhrzeit nicht zur Kontrolle von Zeitbedingungen oder zur Bestimmung des zeitlichen Abstandes des Auftretens von Ereignismeldungen genutzt werden, sondern auf Basis der definierten Nachfolger-Relation nur zur Evaluierung von deren Reihenfolge.

Physikalische Uhren hingegen erlauben die Herstellung eines Bezuges zwischen dem Wert eines Zeitstempels und der Realzeit und damit auch - innerhalb der durch Synchronisationsgenauigkeit und Auflösung gesetzten Grenzen - die Bestimmung des zeitlichen Abstandes von Ereignissen [102].

In einem verteilten System verfügen die einzelnen Stationen in der Regel über lokale Uhren. Daraus resultiert die Notwendigkeit der Synchronisation dieser lokalen Uhren, für die in der Literatur mehrere Möglichkeiten vorgeschlagen werden (z.B. [103]). Eine der wesentlichen Anforderungen dabei ist, daß der Lauf der Uhren streng monoton ist. Im Falle einer festgestellten Abweichung einer lokalen Uhrzeit von der globalen Uhrzeit müssen bezüglich der Korrektur besondere Vorkehrungen getroffen werden, um daraus potentiell resultierende Folgefehler in der verteilten Applikation zu verhindern. Diese Vorkehrungen bestehen in der Resynchronisation der lokalen Uhr, die auf verschiedene Arten geschehen kann, und in der Bereitstellung von Informationen über die aktuelle Güte der Uhrzeitsynchronisation.

Physikalische Uhren und der Synchronisationsalgorithmus müssen in der Lage sein, die von *Lamport* [100] geforderten Bedingungen zu erfüllen, die die Gleichmäßigkeit des Laufes (Bedingung *PU 1*) und die maximale Abweichung zweier beliebiger Uhren (Bedingung *PU 2*) betreffen:

Bedingung PU 1: *Sei $C_i(t)$ eine kontinuierliche, differenzierbare Funktion und repräsentiere sie die Uhrzeit der Uhr in der Station i zum Zeitpunkt t (in Realzeit). Dann existiert eine Konstante $\varkappa \ll 1$ so daß für die Uhren aller Stationen gilt:*

$$\left| \frac{dC_i(t)}{dt} \right| < \varkappa \qquad (1)$$

Bedingung PU 2: *Sei $C_i(t)$ wie in PU 1. Es existiert eine Konstante ε, so daß für alle i,j und t gilt:*

$$\left| C_i(t) - C_j(t) \right| < \varepsilon \qquad (2)$$

Der Wert für die in Bedingung *PU 2* geforderte Konstante ε ist ebenso wie die benötigte Granularität G der Uhr anwendungsabhängig und liegt bei Feldkommunikationssystemen in der Regel im Bereich weniger Millisekunden. So muß ε beispielsweise bei der Regelung von Antrieben über das Kommunikationssystem im Bereich von 1 bis 3 ms [104] und bei der dezentralen Erfassung und Zeitstempelung von Alarmen bei 20 ms [105] liegen. Die Granularität muß dafür mindestens die doppelte zeitliche Auflösung bieten [102].

Ein Zeitstempel $\widehat{C}(t)$ einer Uhr zu einem beliebigen Zeitpunkt t weist den Charakter ganzzahliger Werte auf und wird durch Abschneiden von kontinuierlicher Zeitinformation wie folgt gebildet:

$$\widehat{C} = \left\lfloor \frac{t}{G} \right\rfloor \qquad (3)$$

Im Zusammenhang mit dem Synchronisationsverfahren an sich wird gefordert, daß sich dieses durch einen niedrigen Bedarf an Ressourcen auszeichnen soll.

3.2 Anforderungen bezüglich der Unterstützung verteilter Systeme in den prozeßnahen Bereichen

Die vernetzten, dezentralen Automatisierungskomponenten bilden zusammen ein verteiltes System. Die sich aus dieser Aufgabe ergebenden Anforderungen an das Kommunikationssystem stellen sich wie folgt dar:

ANFORDERUNG ***VS-1: Benutzergesteuerte Prioritätenvergabe***

In Echtzeit-Feldkommunikationssystemen muß es möglich sein, die Dringlichkeit von Aufträgen durch den Benutzer mittels Prioritäten festlegen zu lassen. Die Vergabe einer Priorität soll sich im Kommunikationssystem auf die Reihenfolge der Ausführung der Kommunikationsaufträge auswirken.

ANFORDERUNG ***VS-2: Synchronisation von Applikationen***

Applikationen werden in einem verteilten Echtzeitsystem parallel ausgeführt. Häufig sind durch den Kontext der globalen Applikation kausale Abhängigkeiten zwischen den lokalen Applikationen gegeben. Dies führt ebenso zu der Notwendigkeit der Synchronisation der lokalen Applikationen, wie deren mögliches Konkurrenzverhalten bzgl. gemeinsam genutzter Betriebsmittel[1] [15, 96]. Die Synchronisation der verteilten Applikationen ist durch das Kommunikationssystem zu unterstützen.

ANFORDERUNG ***VS-3: Scheduling des Kommunikationssystems***

Das Kommunikationssystem stellt für das verteilte System eine gemeinsam benutzte Ressource dar. Von zentraler Bedeutung ist, daß der Zugriff auf diese Ressource so gesteuert wird, daß die Echtzeitdienste korrekt abgearbeitet werden können.

1. Betriebsmittel sind hier in einem weiteren Sinne zu verstehen; sie schließen also z.B. gemeinsam genutzte Bereiche des Arbeitsraumes von Robotern o.ä. mit ein.

Das Scheduling des Kommunikationssystems kann vor oder während der Laufzeit des Systems vorgenommen werden. Zudem existieren Mischformen des Schedulings. Die Problematik eines optimalen Schedulings (vgl. dazu z.B. [53]) läßt es sinnvoll erscheinen, die Echtzeitdienste fest im Rahmen der Konfigurierung, alle anderen Dienste aber dynamisch zur Laufzeit des Systems einzuplanen.

ANFORDERUNG VS-4: Breites Spektrum an PDU-Sequenzen

Das Kommunikationssystem sollte in der Lage sein, die vier nachfolgend vorgestellten und in Bild 19 veranschaulichten PDU-Sequenzen abhandeln zu können:

Typ 1: Senden einer Nachricht („Request") an eine Station ohne Quittung oder Antwort.

Typ 2: Senden einer Nachricht („Request") im Multicast ohne Quittung oder Antwort.

Typ 3: Senden einer Nachricht („Request") an eine Station mit Quittung oder Antwort („Response") von dieser Station.

Typ 4: Senden einer Nachricht („Request") an eine Station; die Quittung oder Antwort („Response") wird an alle Stationen übermittelt.

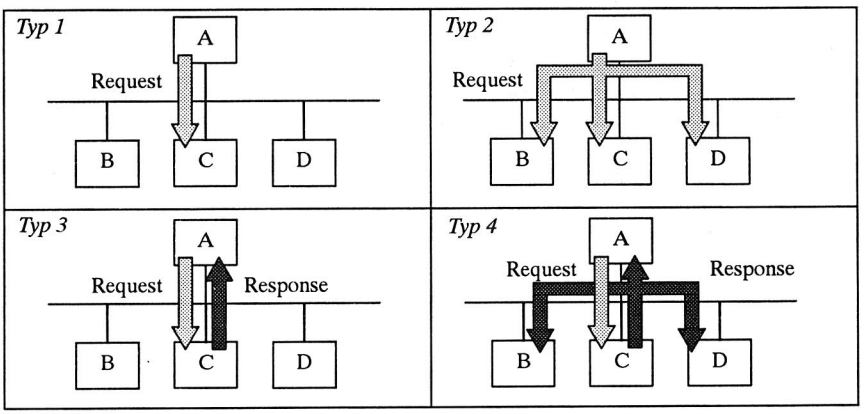


Bild 19: Geforderte PDU-Sequenzen

Die Typen 1 - 3 entsprechen den auch in anderen Kommunikationssystemen gebräuchlichen Kommunikationsbeziehungen zwischen Stationen. Der vierte Typ hingegen ist spezifisch für das Betreiben einer verteilten Datenbasis [106].

ANFORDERUNG VS-5: Autonomie

Die verteilten Applikationen kooperieren in der Regel im Kontext einer globalen Applikation. Dabei besteht die Forderung, daß ein Großteil der jeweils lokalen Applikationen autonom abgewickelt wird. Insbesondere wird auch im Zusammenhang mit der Fehlertoleranz des Gesamtsystems gefordert, daß die lokalen Systeme in der Lage sind, ihre Applikation ohne Kontrolle durch übergeordnete Systeme auszuführen oder nach dem Auftreten eines Fehlers zumindest eigenständig einen sicheren Zustand einzunehmen [19].

ANFORDERUNG VS-6: Komfortable Kommunikationsdienste

Aus Sicht der Anwender stellt sich die Forderung nach leistungsfähigen und einfach zu nutzenden Kommunikationsdiensten und -protokollen. Das Einsatzgebiet von Feldkommunikationssyste-

men in der rechnerintegrierten Fertigung läßt es sinnvoll erscheinen, dabei eine an MMS angelehnte Funktionalität zu fordern. Die darin spezifizierte Funktionalität und die verwendeten Konzepte finden allgemeine Anerkennung, müssen aber für den Einsatz im prozeßnahen Bereich an dessen besonderen ökonomischen und technischen Anforderungen angepaßt werden. Diese Anpassung beinhaltet insbesondere auch die Betrachtung zeitbehafteter Daten [107].

ANFORDERUNG VS-7: Berücksichtigung von ereignis- und zustandsgesteuerter Kommunikation

Prinzipiell wird zwischen zustands- und ereignisgesteuerter Kommunikation unterschieden. Bei der zustandsgesteuerten Kommunikation werden zustandsbeschreibende Variable unabhängig davon, ob sich ihr Wert seit dem letzten Austausch geändert hat oder nicht, periodisch über das Kommunikationssystem übertragen. Das Kommunikationsverhalten ist folglich vorhersehbar und deterministisch. Die Systemkonfiguration verlangt aber nach anwendungsabhängig korrekter Festlegung der Parameterwerte, um den Verlust von kurzzeitigen Ereignissen oder das Verschmelzen von zwei kurz aufeinanderfolgenden Ereignissen auszuschließen. Zudem läßt sich keine genaue Aussage über die Dauer des Ereignisses machen.

Demgegenüber findet bei der ereignisgesteuerten Kommunikation nur dann ein Austausch von Daten statt, wenn sich ein Zustand geändert hat. Das Verhalten des Systems ist durch eine im Normalbetrieb niedrige Grundlast gekennzeichnet. Als problematisch erweist sich, daß kein Determinismus bezüglich des Auftretens von Kommunikationsanforderungen angegeben werden kann und Alarme in technischen Systemen in der Regel zu einem lawinenartigen Kommunikationsaufkommen führen. Das Verhalten von ereignis- bzw. zustandsgesteuerten Systemen nach Eintritt eines Ereignisses, modelliert durch den Wechsel des Wertes von „low" nach „high", ist in Bild 20 beispielhaft dargestellt. Die Zeitachsen sind in den vier Diagrammen gleich. Dabei wird bei zustandsgesteuerten Systemen weiter zwischen Verfahren mit und ohne Zwischenspeicherung von Ereignissen differenziert.

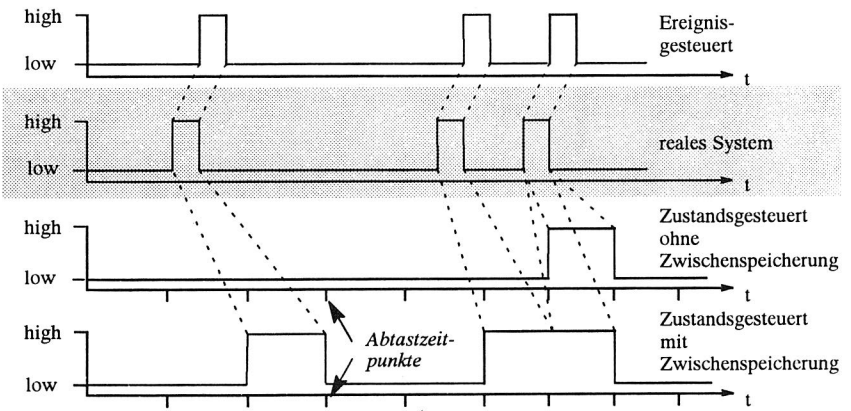


Bild 20: Vergleich zustands- und ereignisgesteuerter Systeme

Weder der ereignisgesteuerte noch der zustandsgesteuerte Ansatz kann allein die Kommunikationsbedürfnisse der Applikationen im prozeßnahen Bereich der rechnerintegrierten Produktion

befriedigen. Aus diesem Grund ist eine Kombination von beiden Kommunikationstypen erforderlich, wobei für beide Typen Echtzeitdienste vorzusehen sind.

ANFORDERUNG VS-8: Unterstützung von Replikationsmechanismen für Daten

Die Verwaltung verteilter Kopien eines Datenwertes trägt maßgeblich zur Fehlertoleranz des Systems beim kurzzeitigen Ausfall einzelner Komponenten bei. Zudem ermöglicht sie in allen Stationen einen schnellen lokalen Zugriff auf die diesem Mechanismus unterworfenen Daten. Prinzipiell kann die Verwaltung replizierter Daten auch durch die Anwendungsinstanzen erfolgen. Die Übernahme dieser Aufgaben durch das Kommunikationssystem bietet jedoch einige Vorteile. Dazu gehören insbesondere:

- Die Anwendungsinstanzen benötigen kein Wissen über die Anzahl und den Ort der Speicherung der Kopien.
- Auf Ebene der Anwendungen muß kein zusätzliches Protokoll existieren [15].

Feldkommunikationssysteme sollen Replikationsmechanismen unterstützen. Es besteht dabei ein Zusammenhang zu den Datenkonsistenzattributen, mit deren Hilfe auf die Gültigkeit der Daten geschlossen werden kann.

3.3 Anforderungen bezüglich der Multimedia-Integration

Die Integration von Multimedia-Diensten führt zu einer Reihe besonderer Anforderungen an das Feldkommunikationssystem. Die Anforderung bezüglich der Übertragung kontinuierlicher Medien werden weitgehend durch die bereits dargestellten Anforderungen bzgl. der Abwicklung von Echtzeitkommunikation abgedeckt, da es sich dabei um einen periodischen Datenaustausch unter Echtzeitbedingungen handelt. Die Übertragung diskreter Medien, z.B. von Standbildern, stellt keine besonderen zeitlichen Anforderungen, sondern kann durch aperiodische Kommunikationsdienste erbracht werden. Neben den Mechanismen zum Austausch der Daten müssen auch komfortable Dienste als Schnittstelle zu diesen bereitgestellt werden.

ANFORDERUNG MM-1: Übertragung kontinuierlicher und diskreter Medien

Gemäß der Definition von Multimedia-Systemen (vgl. Kapitel 2.5) beinhalten diese mindestens ein kontinuierliches und ein diskretes Medium. Das Kommunikationssystem, das zur Erfüllung der in einem verteilten Multimedia-System auftretenden Kommunikationsanforderungen eingesetzt wird, muß somit in der Lage sein, beide Arten von Medien zu übertragen [58].

Bei der Übertragung von diskreten Medien müssen dabei in der Regel keine besonderen zeitlichen Randbedingungen eingehalten werden. Es wird jedoch ein Mechanismus zur Segmentierung und Reassemblierung der typischerweise großen und damit nicht in einem einzigen Datenpaket übertragbaren Datenblöcke, z.B. von kodierten Bildern, benötigt.

Für die Übertragung kontinuierlicher Medien müssen die maximale Verzögerungszeit und der Jitter in der Übertragungszeit begrenzt sein (isochrone Datenübertragung). Die Bereitstellung eines derartigen Mechanismus zur Datenübertragung erlaubt durch Reservierung von dem Anwendungszweck entsprechend ausreichender Bandbreite einen Austausch von Daten für alle die Typen von Datenströmen, bei denen eine obere Grenze für den Bedarf an Übertragungsbandbreite angegeben werden kann (z.B. MPEG, H.261, ADPCM) und ist damit hinreichend. Die Übertragung der Daten von der Quellstation an eine Gruppe von als Senken fungierenden Stationen ist zu unterstützen.

Das Kommunikationssystem soll die dynamische Reservierung von Bandbreite unterstützen, um zeitweise für die Übertragung kontinuierlicher Medien nicht genutzte Ressourcen für andere An-

wendungen zur Verfügung stellen zu können. Sowohl in der Quell- als auch in der Zielstation sollen die Informationen durch das Kommunikationssystem möglichst ohne Verzögerungen weitergeleitet werden. Die Übertragung von Daten von einer Quellstation an mehrere Zielstationen ist vorzusehen.

Als besondere Randbedingung ist in Feldkommunikationssystemen zu beachten, daß die Übertragung von Daten, die Informationen kontinuierlicher oder diskreter Medien repräsentieren, den Echtzeitanforderungen unterliegenden Produktivdatenaustausch nicht behindern darf und deshalb mit vergleichsweise niedrigerer Priorität durchzuführen ist.

ANFORDERUNG MM-2: Unterstützung der Steuerung von Aufnahmesystemen durch das Kommunikationsprotokoll

Das Kommunikationsprotokoll der Anwendungsschicht soll in der Lage sein, auf Basis einer Abstraktion von der Ausprägung des realen Gerätes, Aufnahmesysteme für multimediale Daten, durch die Bevorratung entsprechender Objekte und darauf wirkender Dienste zu unterstützen [59, 108]. Dazu gehören operative Dienste wie das Starten und Stoppen der Aufnahme, das Setzen von Positionsparametern bei nicht ortsfesten Aufnahmesystemen und das Vornehmen von Einstellungen am Gerät selbst, sowie administrative Dienste, z.B. die Anforderung der Kontrolle über ein Gerät oder die Abfrage von dessem aktuellen Status. Die angebotenen Dienste müssen auch geeignet sein, die besonderen Eigenschaften spezieller Kodierungsverfahren sowie die Skalierung der Datenquelle anzusprechen.

ANFORDERUNG MM-3: Synchronisation von Multimedia-Datenströmen

In größeren verteilten Applikationen kann es vorkommen, daß Multimedia-Datenströme mehrerer Quellen vor der Weiterverarbeitung beim Empfänger synchronisiert werden müssen. Das Kommunikationssystem soll diese Synchronisation zumindest ermöglichen und – wenn möglich – auch durch adäquate Mechanismen unterstützen [60].

3.4 Vorstellung und Bewertung ausgewählter Feldkommunikationssysteme

In diesem Abschnitt werden zwei verfügbare relevante Feldkommunikationssysteme vorgestellt und bezüglich der Erfüllung der in den Kapiteln 3.1 und 3.3 dargestellten Forderungen bewertet. Zu den ausgewählten Feldkommunikationssystemen, deren grundlegende Charakteristika in Tabelle 3 dargestellt sind, gehören:

- PROFIBUS, ein in Deutschland genormtes System, das zusammen mit der vorgeschlagenen Normerweiterung PROFIBUS-DP vorgestellt wird, und
- FIP, ein in Frankreich normiertes Feldkommunikationssystem.

Die Entwicklungen an dem durch die IEC international zu standardisierenden Feldkommunikationssystem werden voraussichtlich nicht vor 1997 abgeschlossen sein. Aus diesem Grunde entfällt hier eine Betrachtung dieses Systems, das ebenso wie FIP und PROFIBUS als Feldbus für eine Vielzahl von Anwendungsfeldern konzipiert wird. Systeme mit einem dedizierten Anwendungsbereich wie z.B. INTERBUS-S, CAN oder ASI (vgl. auch 2.3.4) werden ob ihres streng eingegrenzten Anwendungsbereiches hier nicht näher betrachtet. Diese Systeme werden im Regelfall folgerichtig als Ergänzung zu Feldkommunikationssystemen in eingesetzt.

Das auf neueren Forschungs- und Entwicklungsarbeiten u.a. des Lehrstuhls für Fertigungsautomatisierung und Produktionssystematik (FAPS) beruhende OLCHFA-System, bei dessen Ausle-

gung eine Teilmenge der dargestellten Anforderungen bzgl. der Unterstützung von Echtzeitkommunikation bereits eingearbeitet wurde, wird in Kapitel 4.1 vorgestellt und bewertet.

Charakteristikum	*PROFIBUS*	*FIP*
Datenübertragungsrate	9,6 - 500 kbps; DP: 1,5 Mbps	31,25 kbps; 1 Mbps; 2,5 Mbps
Max. Ausdehnung unter Verwendung von Leitungsverstärkern (bei Datenrate)	9,6 kbps: 4800 m 500 kbps: 800 m 1500 kbps: 200 m	2000 m (mit Repeatern)
max. Teilnehmerzahl	122	255
Zugriffsverfahren	hybrid: Token Passing, Polling	zentral: Delegated Token
Normung	DIN 19245	UTE C-46-6xx

Tabelle 3: Charakteristika ausgewählter Feldkommunikationssysteme

Der Schwerpunkt der nachfolgenden Vorstellungen der Systeme liegt auf der Erläuterung der wesentlichen Basisprinzipien, auf denen die Protokolle der Sicherungs- und Anwendungsschicht beruhen. Die in die Sicherungsschicht einzuordnenden Protokolle erlauben dabei bis auf wenige Ausnahmen, z.B. bei der drahtlosen Übertragung von Daten nach dem Frequency-Hopping Verfahren, eine Abstraktion von der Ausprägung der Bitübertragungsschicht, so daß für diese in den nachfolgenden Ausführungen eine kurze Betrachtung ausreichend ist.

3.4.1 Feldkommunikationssystem PROFIBUS

Im Rahmen des vom Bundesministerium für Forschung und Technologie (BMFT) geförderten Verbundprojektes *Feldbus* schlossen sich insgesamt vierzehn Hersteller und fünf technisch- wissenschaftliche Institute zusammen, um das von den Firmen Siemens, Bosch und Klöckner-Moeller ausgearbeitete Feldbuskonzept PROFIBUS (*PROcess FIeld BUS*) in einen Standard umzusetzen [109]. Bei der Entwicklung dieses Standards ist besonders darauf geachtet worden, daß verschiedene Anwendungsgebiete in der Fertigungs-, Prozeß- und Gebäudeautomatisierung abgedeckt werden können und eine Durchgängigkeit zu übergeordneten Netzwerken bei insgesamt niedrigen Anschlußkosten für einfache Systemkomponenten besteht. Die Verabschiedung als deutsche Norm DIN 19245 Teil 1 und Teil 2 ([110] bzw. [111]) erfolgte 1991. Bestrebungen, PROFIBUS auch im Bereich der dezentralen Ein-/Ausgabe kostengünstig einsetzen zu können führten 1994 zu der Schaffung des Teils 3 „Dezentrale Peripherie (DP)" der PROFIBUS Norm ([112]). In Hinblick auf Anwendungen in der Verfahrenstechnik wird PROFIBUS derzeit um die Variante PROFIBUS-PA (Prozeßautomatisierung) erweitert [113].

Zur eindeutigen Unterscheidung der betrachteten Varianten werden Aussagen, die für das System allgemeingültig sind, als für „PROFIBUS" geltend, solche die nur für die ursprüngliche Definition des PROFIBUS gemäß DIN 19245 Teil 1 und 2 zutreffen, als für „PROFIBUS-FMS", und solche, die sich auf PROFIBUS gemäß DIN 19245 Teil 1 und Teil 3 beziehen, als für „PROFIBUS-DP" geltend erwähnt.

PROFIBUS-FMS ist ein Feldkommunikationssystem mit Protokollen für die Schichten 1, 2 und 7 des ISO-OSI Basisreferenzmodells. Bei PROFIBUS-DP sind nur die Schichten 1 und 2 ausgeprägt. Die Kommunikationsfunktionalität wird in PROFIBUS-DP über eine definierte Dienstschnittstelle zur Verfügung gestellt. Für beide Systeme sind zudem, wie in Bild 21 dargestellt, Funktionen des Managements für die ausgeprägten Schichten definiert.

	ALI (Application Layer Interface)	Benutzerschnittstelle DDLM	
7	FMS / LLI		FMA-7
3 - 6			
2	FDL	FDL	FMA-1/2
1	(LWL) RS-485 (EEX-I)	RS-485	
	PROFIBUS	*PROFIBUS-DP*	*Management*

Bild 21: Protokollarchitekturen von PROFIBUS und PROFIBUS-DP

Eigenschaften der Bitübertragungsschicht

PROFIBUS basiert auf dem Standard RS-485 der EIA. Die Verkabelung erfolgt busförmig. Bezüglich der Übertragungsrate sind die Alternativen 9,6 kbps, 19,2 kbps, 93,75 kbps, 187,5 kbps und 500 kbps vorgesehen. Die Normerweiterung PROFIBUS-DP erlaubt zusätzlich eine Datenübertragungsrate von 1.500 kbps. Die maximale Leitungslänge ist abhängig von der verwendeten Übertragungsgeschwindigkeit und liegt zwischen 200 m und 4800 m. Als Medium wird eine verdrillte Zweidrahtleitung eingesetzt. Daneben besteht auch die Möglichkeit, Lichtwellenleiter (LWL) als alternative Übertragungstechnik einzusetzen. Die Kodierung der übertragenen Bits erfolgt nach dem NRZ-Verfahren, wobei die einzelnen Zeichen als UART-Zeichen übertragen werden. Die Übertragung an sich erfolgt asynchron und schlupffrei. Die Verwendung von Zeichenwiederholung und einer Blockprüfsumme führt zu einer Hamming-Distanz der Telegramme von HD=4.

Pro Bussegment können bis zu 32 Teilnehmer angeschlossen werden. Der Einsatz von Leitungsverstärkern (Repeatern) erlaubt die Kopplung von Bussegmenten. Zwischen zwei Stationen dürfen dabei maximal drei Leitungsverstärker angeordnet sein, die als Busteilnehmer zählen. Damit können in derartigen Konfigurationen insgesamt bis zu 122 Teilnehmerstationen zusätzlich zu den Leitungsverstärkern angeschlossen werden [32].

Für den Einsatz in explosionsgefährdeten Bereichen existiert eine eigensichere Variante von PROFIBUS. Diese basiert in der Bitübertragungsschicht auf der IEC-Norm 1158-2. Für die Kodierung der Zeichen wird das Manchester-II Verfahren eingesetzt. Die Datenübertragungsrate liegt bei 31,25 kbps [114].

Eigenschaften der Sicherungsschicht

PROFIBUS basiert auf dem Master-Slave Verfahren mit einem hybriden Medienzugriffsverfahren. Master sind aktive Stationen, die untereinander die Buszugangsberechtigung in Form eines Tokens zyklisch austauschen (*Token-Passing*) und Slaves sind passive Stationen, die nur auf die Anforderung eines Masters hin (*Polling*) Daten auf dem Bus senden dürfen (vgl. Bild 22).

Die Sicherungsschicht, im PROFIBUS-System als FDL (Fieldbus Data Link) bezeichnet, ist für die Steuerung des Zugriffs auf das Medium und für eine geregelte Abwicklung der Kommunikation verantwortlich. Sie beinhaltet vollständig die Tokenverwaltung, Initialisierung und Verwaltung des logischen Ringes, Fehlerüberwachung sowie Überwachung der Übertragungsdienste.

Das Protokoll der Sicherungsschicht von PROFIBUS bietet neben der Unterstützung des azyklischen Datenverkehrs durch die Dienste *Send Data with Acknowledge (SDA), Send Data with*

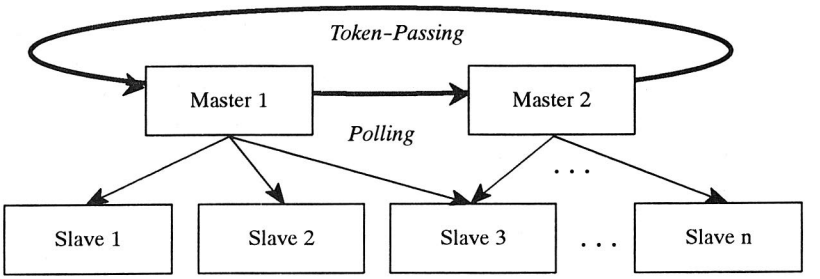


Bild 22: Hybrides Medienzugriffsverfahren bei PROFIBUS

no Acknowledge (SDN) und *Send and Request Data with Reply (SRD)* auch Unterstützung für einen zyklischen Nachrichtenaustausch (*Cyclic Send and Request Data with Reply, CSRD)*. Dieser wird nur einmal durch eine Dienstanforderung der Anwendungsschicht angestoßen und läuft danach weitgehend autonom und asynchron zu den Anwendungsprozessen in der Sicherungsschicht ab. Die zu übertragenden Daten werden in den erzeugenden und verbrauchenden Stationen in einem Pufferspeicher gehalten. Die Reihenfolge dieser Abfrage wird bei den aktiven Teilnehmern in einer sogenannten Polling-Liste hinterlegt. Während der Ausführung des zyklischen Datenverkehrs können Listeneinträge zur Laufzeit gesperrt und wieder freigegeben werden. Zudem besteht die Möglichkeit, die aktive Polling-Liste durch eine andere zu ersetzen. Die Verlagerung der Polling-Prozedur in die Sicherungsschicht, verbunden mit der verteilten Speicherung des Abbildes, bedeutet eine erhebliche Geschwindigkeitserhöhung für zeitkritische Anwendungen, da der Anwendung die aktuellen Daten jeweils lokal, also ohne Buszugriff, zur Verfügung stehen. Das Senden von Daten an mehrere Stationen gleichzeitig (*Multicast*) kann durch die Verwendung des Dienstes SDN erwirkt werden.

Die Funktionalität von PROFIBUS-DP wird über das DDML oberhalb des Protokolls der Sicherungsschicht angeboten. Die Kommunikation erfolgt verbindungslos zwischen Mastern der Klassen 1 (reguläre Masterstationen) und der Klasse 2 (Programmier- bzw. Managementgeräte) bzw. zwischen derartigen Mastern und Slaves. Dabei ist wiederum ein Multicast möglich. Zwischen Mastern werden nur administrative Dienste abgewickelt. Dazu gehören das Lesen von Diagnoseinformationen, das Herauf- und Herunterladen von Datensegmenten sowie das Aktivieren und Deaktivieren von Parametersätzen. Für die Master-Slave Kommunikation steht daneben ein Dienst für den Produktivdatenverkehr zur Verfügung, der es gestattet, aktualisierte Werte für Ausgänge der Slave-Stationen vom Master an diese zu übermitteln und in der Quittung die dort vorliegenden aktuellen Werte der Eingänge zu lesen. Für das Setzen von Ausgängen und das Lesen von Eingängen existiert ein Synchronisationsmechanismus, der es erlaubt, die Eingangsinformationen in allen angeschlossenen Slaves gleichzeitig zu lesen und lokal „einzufrieren" sowie Ausgangsinformationen in allen Slaves gleichzeitig freizuschalten. Der Austausch der Daten zwischen Mastern der Klasse 1 und Slaves wird zyklisch unter Kontrolle des Schedulers vom Master abgewickelt. Eine einfache Kodierung der Daten wird vorgenommen. Bezüglich des Systemmanagements stehen in PROFIBUS lokal und entfernt wirkende Funktionen zur Verfügung.

Eigenschaften der Anwendungsschicht

Die Anwendungsschicht des PROFIBUS-FMS besteht aus zwei Teilschichten: Zum einen ist dort als Schnittstelle zur Schicht 2 das *Lower Layer Interface* (LLI) und zum anderen das eigentliche anwendungsunterstützende Protokoll *Fieldbus Message Specification* (FMS) definiert.

Zu den Aufgaben des LLI gehört insbesondere die Abbildung der FMS-Dienste auf die FDL-Dienste und umgekehrt. Daneben ist es zuständig für das Management der Verbindung zur Nachbildung notwendiger Funktionalität der Protokolle der nicht ausgeprägten Schichten 3 bis 6, wie zum Beispiel Flußkontrolle oder Verbindungsauf- und -abbau. Als Verbindungsarten stehen hier die Alternativen *Master-Slave zyklisch/azyklisch mit/ohne Slave Initiative* und *Master-Master azyklisch* zur Verfügung. In letzterem Fall emuliert einer der Master den Slave.

Das FMS-Protokoll orientiert sich am MMS-Protokoll und stellt dem Anwender eine funktionale Untermenge der darin definierten Dienste und Protokolle zur Verfügung. Dabei wurde insbesondere auf eine Anpassung der Funktionalität an die begrenzte Verarbeitungskapazität einfacher Feldgeräte geachtet. Neu hinzugekommen ist das Prinzip der vordefinierten *Kommunikationsbeziehungen* zwischen Stationen, sowie die Verwaltung der Kommunikationsobjekte mit Hilfe von *Objektverzeichnissen*.

Im FMS-Protokoll finden sich an die Bedürfnisse von Feldkommunikationssystemen angepaßte MMS-Dienste und Objekte für

- die Verwaltung von Verbindungen (*Context Management*),
- das Übertragen von Datenblöcken (*Domain Management*),
- die Unterstützung des virtuellen Feldgerätes (*VFD Support*),
- den Zugriff auf Variable (*Variable Access*),
- die Behandlung von Ereignissen (*Event Management*)

sowie zusätzliche Dienste und Objekte für

- die Verwaltung von Objektverzeichnissen (*OV Management*).

In PROFIBUS-FMS ist die Kodierung und Dekodierung der FMS-PDUs in das FMS Protokoll aufgenommen worden. Eine Übersicht über die in FMS spezifizierten Dienste und Dienstgruppen wird z.B. von *Bender* [32] gegeben. Das Netzmanagement der Schicht 7 von PROFIBUS-FMS stellt Dienste aus den Bereichen *Kontextmanagement*, *Konfigurationsmanagement* und *Fehlermanagement* zur Verfügung, die teils auf die lokale und teils auf eine entfernte Station wirken.

3.4.2 Bewertung von PROFIBUS

Nachfolgend wird die qualitative Bewertung des Feldkommunikationssystems PROFIBUS bezüglich der Erfüllung der erarbeiteten Anforderungen dargestellt.

ANFORDERUNG EZ-1: Rechtzeitigkeit

PROFIBUS-FMS garantiert, unabhängig davon, ob die zur Steuerung der Weitergabe der Sendeberechtigung verwendete Zielzeit für den einmaligen Umlauf eines Tokens – die *Target-Token-Rotation-Time* – bereits überschritten wurde oder nicht, die Abwicklung mindestens eines hochprioren Dienstes pro Station und Tokenumlauf. Damit wird gewährleistet, daß mindestens eine Nachricht pro Tokenumlauf gesendet werden kann. Das Verfahren zur Tokenverwaltung ist dabei so ausgelegt, daß bei korrekter Parameterisierung des Systems die konfigurierte Tokenumlaufzeit eingehalten wird.

Das Token wird von einer Masterstation aber auch dann weitergeben, obwohl noch Tokenhaltezeit zur Verfügung steht, wenn alle anliegenden Übertragungsanforderungen der Station abgearbeitet sind. Dies führt zu einer vorzeitigen Ankunft des Tokens bei der logisch nachfolgenden Masterstation. Dadurch entsteht ein Jitter in der Periodizität der Tokenankunftszeit, der sich auf alle Klassen von Übertragungsanforderungen auswirkt. Generell können für Kommunikationsanforderungen mittlerer oder niedriger Priorität keine Garantien gemacht werden. Damit besteht kein Determinismus für periodischen Datenverkehr.

In PROFIBUS-DP werden die Produktivdienste zwischen Master und den zugeordneten Slaves grundsätzlich hochprior und zyklisch abgewickelt. Damit läßt sich auch in Multi-Master-Konfigurationen bei korrekter Festlegung der Parameter des Systems eine obere Schranke für die Wartezeit auf Ausführung periodischer Übertragungsanforderungen garantieren. Für aperiodische Kommunikationsanforderungen können keine Zeitschranken angegeben werden.

ANFORDERUNG EZ-2: Berücksichtigung von Echtzeit- und Nicht-Echtzeit Kommunikationsanforderungen

PROFIBUS unterstützt sowohl Echtzeit- als auch Nicht-Echtzeit Kommunikationsdienste. Erstere stehen der Anwendung nur mit den hochprioren Diensten zur Verfügung. Bei dem zu PROFIBUS-FMS gehörigen zyklischen Datenaustausch ist keine Garantie für eine deterministische Dienstausführung in einem festen Zeitraster gegeben, denn hochpriore Nachrichten und – je nach Betriebszustand – auch niederpriore Nachrichten können die Abarbeitung der Pollingliste verzögern. Zudem kann durch das Prinzip der Tokenweitergabe (s.o.) ein Jitter in der Periodizität der Abarbeitung der Pollingliste auftreten.

Bei korrekter Parameterisierung des Systems kann PROFIBUS-DP auch in Multi-Master-Konfigurationen Echtzeitbedingungen erfüllen.

ANFORDERUNG EZ-3: Vorhersehbarkeit

Eine Vorhersehbarkeit bzgl. der Ausführungszeitpunkte für Dienstanforderungen gibt es bei PROFIBUS-FMS nur für die hochprioren Dienste. Hier kann durch die Garantie der Abwicklung mindestens eines hochprioren Auftrages pro Tokenumlauf eine maximale Zeitschranke für die Übertragung der Anforderung angegeben werden. Die Einlastung vieler hochpriorer Anforderungen führt zu Verzögerung der Dienstabwicklung von mittel- und niederprioren Kommunikationsanforderungen.

Das Verhalten von PROFIBUS-DP bezüglich der Abwicklung periodischer Kommunikationsvorgänge ist vorhersehbar. Die Kommunikation wird durch den bzw. die Master im System gesteuert und beinhaltet den automatischen zyklischen Austausch von Daten zwischen Master und Slaves.

ANFORDERUNG EZ-4: Zuverlässigkeit

PROFIBUS-FMS ist als Multi-Master-System konzipiert. Damit besteht die Möglichkeit, auf Ebene der Applikation Redundanz vorzusehen. Redundanz wird von PROFIBUS auch bezüglich des Übertragungsmediums unterstützt. In PROFIBUS-DP wird die Redundanz auf Ebene der Applikation nicht unterstützt, da eine Zuordnung der Slave-Stationen zu je genau einem Master existiert. Festgestellte Fehlerzustände einer Station können von dieser über das Management von PROFIBUS an andere Stationen gemeldet werden.

ANFORDERUNG EZ-5: Verarbeitung von Echtzeit-Prozeßvariablen

Weder PROFIBUS-FMS noch PROFIBUS-DP beinhalten Mechanismen zur Behandlung zeitbehafteter Daten. PROFIBUS-FMS beinhaltet jedoch Kodierungsmechanismen zur Verarbeitung

der Datentypen *Date*, *Time-of-Day* und *Time-Difference*, die von PROFIBUS-DP nicht angeboten werden. Die maximale Auflösung beträgt in allen Fällen eine Millisekunde. Datenkonsistenzattribute werden weder von PROFIBUS noch von PROFIBUS-DP unterstützt.

ANFORDERUNG ***EZ-6:* Unterstützung einer verteilten Uhr**

Eine verteilte Uhr wird durch PROFIBUS nicht unterstützt.

ANFORDERUNG ***VS-1: Benutzergesteuerte Prioritätenvergabe***

An der Diensteschnittstelle von PROFIBUS-FMS wird nur für unbestätigte Dienste die Wahl zwischen einer hoch- oder niederprioren Dienstausführung angeboten. Alle bestätigten Dienste werden stets mit niedriger Priorität ausgeführt. PROFIBUS-DP bietet an der Benutzerschnittstelle keine Prioritätenvergabe für Dienstanforderungen an.

ANFORDERUNG ***VS-2: Synchronisation von Applikationen***

PROFIBUS bietet keine expliziten Mechanismen für die Synchronisation von Applikationen an. PROFIBUS-DP stellt zur Synchronisation von Applikationen das „Einfrieren" (FREEZE) von Eingangsinformationen und das gleichzeitige Freischalten von Ausgängen (SYNC) zur Verfügung.

ANFORDERUNG ***VS-3: Scheduling des Kommunikationssystems***

Ein Scheduling des Kommunikationssystems findet in PROFIBUS stationsorientiert durch die Festlegung der Tokenhaltezeit für Masterstationen und der Target-Token-Rotation-Time statt. Die Station nutzt die ihr zur Verfügung stehende Übertragungszeit zur Sendung von hoch- und niederprioren Nachrichten. In PROFIBUS-FMS wird zudem die konfigurierte Pollingliste abgearbeitet. Zur Laufzeit des Systems können durch lokale Funktionen des Managements Einträge in der Pollingliste gesperrt und wieder freigegeben werden.

ANFORDERUNG ***VS-4: Breites Spektrum an PDU-Sequenzen***

PROFIBUS-FMS unterstützt auf der Ebene der Sicherungsschicht vier verschiedene Dienste, mit denen die geforderten PDU-Sequenzen der Typen 1 bis 3 abgedeckt werden. Die Typen 1 und 2 können mit Hilfe des SDN-Dienstes durch entsprechendes Setzen der Zieladresse realisiert werden. Die PDU-Sequenz vom Typ 3 kann wahlweise durch den SDA, SRD oder CSRD-Dienst erbracht werden. Das Senden einer Nachricht an eine Station mit nachfolgender Übertragung der Quittung an alle Stationen wird nicht unterstützt. PROFIBUS-DP nutzt nur die Dienste SDN und SRD, mit denen ebenfalls die PDU-Sequenzen 1, 2 und 3 abgewickelt werden können.

ANFORDERUNG ***VS-5: Autonomie***

Das Multi-Master-Prinzip von PROFIBUS erlaubt die Funktion von Teilsystemen auch nach dem Ausfall von Komponenten. Das FMS-Protokoll unterstützt die Steuerung der Abwicklung von Steuerungsprogrammen (Program Invocations) durch Bereitstellung von entsprechenden Diensten. Es sind jedoch nur Master-Stationen oder ausgezeichnete Slaves – die Slaves mit Initiative – befähigt, den Austausch von Informationen zu initiieren. PROFIBUS-FMS sieht ein Ereignismanagement vor, mit dessen Hilfe eine Benachrichtigung über den Eintritt eines Ereignisses an eine oder mehrere Stationen gesendet werden kann.

PROFIBUS-DP stellt als Unterstützungsfunktionen für die Autonomie einer lokalen Station nur die Möglichkeit bereit, daß Slaves das Vorliegen von Diagnose- oder Fehlerinformationen anzeigen können. Der zugeordnete Master kann dann die Diagnoseinformation auslesen.

ANFORDERUNG VS-6: Komfortable Kommunikationsdienste

PROFIBUS-FMS bietet mit dem FMS-Protokoll eine funktionale, auf den Einsatz in der prozeßnahen Kommunikation ausgerichtete, Untermenge des MMS-Protokolls an. Der geforderte Mindestumfang einer FMS-Implementierung beinhaltet nur administrative Dienste. Als problematisch erweist sich, daß das Herunterladen (Download) von Programmen nur zwischen Masterstationen möglich ist.

PROFIBUS-DP bietet eine Schnittstelle zum Zugriff auf die Dienste des Protokolls der Sicherungsschicht an und beinhaltet Funktionalität zum Zugriff auf dezentrale Ein- und Ausgänge sowie administrative Dienste.

ANFORDERUNG VS-7: Berücksichtigung von ereignis- und zustandsgesteuerter Kommunikation

PROFIBUS-FMS unterstützt sowohl die ereignis- als auch die zustandsgesteuerte Kommunikation. Beide Modi werden von der Applikation durch Aufruf entsprechender Dienste initiiert, wobei die zustandsgesteuerte Kommunikation, d.h. die Abarbeitung der Pollingliste, nach dem ersten Aufruf eines Dienstes zum Lesen oder Schreiben einer Variablen autonom abläuft. Der ereignisgesteuerten Übertragung von Alarmen ist in PROFIBUS-FMS dabei die höchste Priorität zugewiesen.

PROFIBUS-DP unterstützt für Master der Klasse 1 nur den zyklischen Austausch von Daten. Master der Klasse 2 können hingegen zyklisch oder azyklisch auf Produktivdaten zugreifen.

ANFORDERUNG VS-8: Unterstützung von Replikationsmechanismen für Daten

Von PROFIBUS-FMS werden Replikationsmechanismen auf Ebene des LLI unterstützt. Die Kopien von Daten, die in Slaves produziert werden, liegen jedoch nur in der Masterstation vor, die über eine existierende Kommunikationsbeziehung den Wert abfragt. Entsprechend liegen Kopien von Werten, die vom Master produziert werden, nur bei genau dem Slave vor, der den Wert konsumiert. Es werden Mechanismen zur Verwaltung einer verteilten Datenbasis damit nur ansatzweise unterstützt.

In PROFIBUS-DP-System liegt in der Masterstation ein Abbild der Zustände der Ein- und Ausgänge der Slaves vor, das zyklisch aktualisiert wird. Die Slaves haben nur Zugriff auf die Werte ihrer lokalen Prozeßvariablen.

ANFORDERUNG MM-1: Übertragung kontinuierlicher und diskreter Medien

Die Übertragung von kontinuierlichen Medien kann mit PROFIBUS-FMS in einer sinnvollen Weise nur unter Verwendung der Pollingliste und nur in einer Mono-Master-Konfiguration realisiert werden, denn die Verwendung hochpriorer Dienste muß den Steuerungsapplikationen zum Absetzen von Alarmmeldungen vorbehalten sein, und für die niederprioren Dienste kann die geforderte Isochronität des Datenaustausches nicht gewährleistet werden. In Multi-Master-Konfigurationen kann während der Zeit, in der das Token einem anderen Master zugeteilt ist, keine Datenübertragung stattfinden, so daß ein isochroner Datenaustausch aufgrund des Fehlens einer verteilten Datenbasis ebenfalls nicht möglich ist.

Bei Verwendung des Polling-Mechanismus besteht die Möglichkeit, vorkonfigurierte Einträge in der Pollingliste auf Anforderung der Applikation hin dynamisch freizuschalten und wieder zu sperren. Nachteilig wirkt sich aus, daß damit die Daten nicht direkt im Broadcast übertragen werden können. Daten diskreter Medien, z.B. Informationen, die ein digitalisiertes Einzelbild reprä-

sentieren, können entweder in Variablen verschlüsselt segmentweise oder als Nutzdaten eines vordefinierten Domains übertragen werden. Die Verzögerung der Weiterleitung der Daten ist in der Quellstation von dem aktuellen Zustand des lokalen Systems abhängig. Als wesentliche Einflußfaktoren sind die Verfügbarkeit des Tokens, die der Station zustehende Tokenhaltezeit, die Anzahl zur Übertragung anstehender hochpriorer Nachrichten und die aktuelle Position in der Abarbeitung der Pollingliste zu nennen. Eine obere Begrenzung für die mögliche Verzögerung kann nicht angegeben werden. Hierbei ist zu beachten, daß die Privilegierung hochpriorer Anforderungen die gewünschte Bevorzugung von Prozeßdatenverkehr gegenüber dem Austausch von zu multimedialen Datenströmen gehörigen Informationen beinhaltet. In der Zielstation werden die Daten im Abbildspeicher des LLI hinterlegt und es wird die Anwendung über deren Eintreffen informiert.

In PROFIBUS- DP können kontinuierliche Medien ebenfalls nur in Mono-Master-Konfigurationen isochron übertragen werden. Dabei kann wiederum keine direkte Übertragung der Daten im Broadcast stattfinden. Die dynamische Einplanung von zyklischer Datenübertragung wird nicht unterstützt. Die Verzögerung der Daten in der Quellstation ist abhängig von der Verfügbarkeit des Tokens und dem internen Zustand der Station und beträgt maximal eine Tokenumlaufzeit. In der Zielstation werden die Daten sofort nach ihrer Ankunft an den Dienstbenutzer weitergegeben.

ANFORDERUNG MM-2: Unterstützung von Aufnahmesystemen durch das Kommunikationsprotokoll

In PROFIBUS-FMS können Aufnahmesysteme als Spezialfall eines VFD betrachtet werden. In diesem Fall müssen jedoch die Funktionen zur Steuerung des Aufnahmesystems in Program Invocations oder in ausgezeichneten Variablen verschlüsselt werden. PROFIBUS-DP bietet für diese Anforderung keine Unterstützungsfunktionen an.

ANFORDERUNG MM-3: Synchronisation von Multimedia-Datenströmen

Bei einer Realisierung des Austausches von zu multimedialen Datenströmen gehörigen Dateneinheiten mittels des zyklischen Austauschmechanismus von PROFIBUS-FMS wird eine implizite Synchronisation durch die Abarbeitungsstrategie der Pollingliste gewährleistet. Wird die Abarbeitung der Pollingliste durch hochpriore Anforderungen verzögert, kann die Synchronisation nicht gewährleistet werden.

PROFIBUS-DP basiert auf zyklischem Datenaustausch. Durch die Abarbeitung der Übertragungsanforderungen in einer festen Reihenfolge wird die Synchronisation von multimedialen Datenströmen implizit unterstützt.

3.4.3 Feldkommunikationssystem FIP

Das FIP-System ist in Frankreich national durch die AFNOR (*Association Française de Normalisation*) genormt. Die Spezifikation geht auf eine Initiative des französischen Ministeriums für Industrie und Technik in den Jahren 1983–1985 zurück. Die Protokolle, die den Austausch von Variablen regeln, und das Protokoll der Schicht 2 für den Nachrichtenaustausch sind bereits normiert. Das Protokoll für den Nachrichtenaustausch auf Schicht 7 ist noch in der Abstimmung.

Wie bei den meisten Feldkommunikationssystemen, z.B. auch beim deutschen PROFIBUS, handelt es sich beim FIP-Protokollstapel um eine reduzierte Schichtenarchitektur. Es sind nur Protokolle für die Bitübertragungs-, die Sicherungs- und die Anwendungsschicht des ISO-OSI Basisreferenzmodells vorgesehen. Schichtübergreifend ist zudem noch das Netzmanagement definiert.

Die Protokollarchitektur ist dabei durch die Ausprägung mehrerer Protokolle für die einzelnen Schichten gekennzeichnet (vgl. Bild 23). Dabei gehört zu jedem Protokoll der Anwendungs-

schicht ein Protokoll der Sicherungsschicht. Für die Übertragung von Nachrichten wurde auf Ebene der Sicherungsschicht das MSG (*Message*)-Protokoll definiert: In der Anwendungsschicht gehört dazu das Protokollpaar MCS (*Message Control Services*) und SubMMS (*Subset MMS*). Für den Austausch von Werten von Variablen über die, als zentralen Bestandteil des FIP-Systems zu betrachtende, verteilte Datenbasis wurde in der Sicherungsschicht das A/P (*Aperiodical/Periodical Variable Exchange*) Protokoll spezifiziert. Dazu gehört das MPS (*Manufacturing Periodical/Aperiodical Services*) Protokoll in der Anwendungsschicht. Zur Koordinierung redundanter Busarbitratoren werden die Protokolle LBAS (*Bus Arbitrator Link Services*) in der Sicherungsschicht und ABAS (*Bus Arbitrator Application Services*) in der Anwendungsschicht eingesetzt.

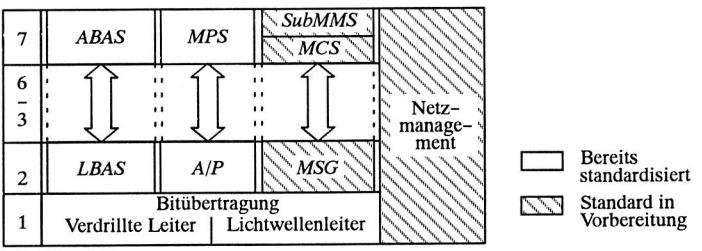


Bild 23: FIP Protokollarchitektur

Eigenschaften der Bitübertragungsschicht

In den der Bitübertragungsschicht zugeordneten Normen werden die Charakteristika der Medien sowie der funktionalen, elektrischen und prozeduralen Eigenschaften für die Übertragung von Bits über das Medium festgelegt. Alternativ wird von FIP die Verwendung von verdrillten Kupferleitungen oder von Lichtwellenleitern unterstützt. Bei der Verwendung verdrillter Leiter stehen drei Datenübertragungsraten zur Auswahl: 31,25 kbps (S1), 1 Mbps (S2) und 2,5 Mbps (S3). Üblicherweise wird dabei die Datenübertragungsrate S2 eingesetzt. Die Alternativen stehen für besondere Ansprüche, wie z.B. die Datenübertragung im explosionsgefährdeten Bereich, zur Verfügung. Die maximale Ausdehnung eines FIP-Feldkommunikationssystems beträgt 2000 m. Dabei können bis zu 255 Stationen gleichzeitig angeschlossen sein. Eingesetzte Repeater dienen dazu, durch Signalverstärkung den Zusammenschluß mehrerer Hauptkabelsegmente zu erlauben.

Eigenschaften der Sicherungsschicht

Die Sicherungsschicht von FIP [115] bietet drei Klassen von Übertragungsdiensten an: Dienste für den Austausch von Busarbitrator-Daten (LBAS), Variablen (A/P) und solche für den Transfer von Nachrichten (MSG). Dabei kommen zwei unterschiedliche Adressierungsmechanismen zum Einsatz. Während die maximal 65536 Variablen über einen systemweit eindeutigen Bezeichner adressiert werden, findet beim Nachrichtenaustausch eine stationsorientierte Adressierung Verwendung.

Die Medienzuteilung wird durch eine ausgezeichnete Station, den *Arbitrator*, zentral geregelt. Die in Vorbereitung befindliche Option, redundante Arbitratoren vorzusehen, wird eine Erhöhung der Verfügbarkeit des Gesamtsystems ermöglichen. Stationen können dabei gleichzeitig sowohl als Arbitrator als auch als normaler Busteilnehmer agieren.

Die zur Verfügung stehende Übertragungskapazität wird in mehrere Teilbereiche, sog. *Kommunikationsfenster*, aufgeteilt, die während der Systemkonfiguration festgelegt werden: ein Fenster für periodische Anforderungen, je ein Fenster für aperiodischen Variablen- und Nachrichtenaustausch sowie ein Synchronisationsfenster (vgl. Bild 24). In dem Fenster für periodischen Verkehr werden Variable und Nachrichten ausgetauscht. Der Arbitrator verfügt dafür über eine vorkonfigurierte Abfragetabelle. In ihr ist neben der Abfragereihenfolge für alle zyklisch zu aktualisierenden Variablen die Zuteilung des Senderechtes für periodischen Nachrichtentransfer festgelegt. Die Zusammenfassung der vier Kommunikationsfenster wird als *Elementarzyklus* bezeichnet. Die Aneinanderreihung von Elementarzyklen wird als *Makrozyklus* bezeichnet.

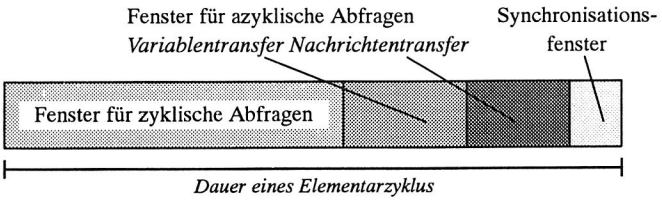


Bild 24: Übertragungsfenster im FIP System

Die Verwaltung der verteilten Datenbasis wird durch das Protokoll der Sicherungsschicht vorgenommen. In dieser werden die Werte von Variablen in verteilten Kopien gehalten. Für jede Variable gibt es genau eine sie erzeugende Station (*Erzeuger*), die auf die Anfrage hin den Wert der Variablen sendet (*Verteilung*) und beliebig viele diese Variable aufnehmende Stationen (*Konsumenten*). Sowohl in der Erzeuger- als auch in den Konsumentenstationen existiert für jede diesem Mechanismus unterliegende Variable ein durch das Protokoll der Sicherungsschicht verwalteter Pufferspeicher. Auf die darin enthaltenen Werte kann von der Anwendungsschicht lesend und schreibend ohne Busaktivität zugegriffen werden. Die Aktualisierung der Werte kann zyklisch oder azyklisch erfolgen.

Zyklischer Datentransfer

Der zyklische Austausch von Werten von Variablen unterliegt der Kontrolle des Busarbitrators. Dieser sendet gemäß der bei ihm vorliegenden Informationen Anfragerahmen für die zyklisch auszutauschenden Variablen, die, wie in Bild 25 beispielhaft dargestellt, von der jeweils produzierenden Station einer Variablen durch das Senden des aktuellen Wertes der Variablen beantwortet werden. Das FIP-Protokoll garantiert für diesen zyklischen *Erzeugung-Verteilung-Konsumierung*-Prozeß einen zeitlichen Determinismus. Die Festlegung der Abfragereihenfolge und der Frequenz, mit der die unterschiedlichen Variablen aktualisiert werden, erfolgt während der Konfigurierung des Systems.

Azyklischer Datentransfer

In den azyklischen Zeitfenstern kann auf Anforderung beliebiger Stationen hin der Austausch von Variablenwerten oder Nachrichten vorgenommen werden. Dazu meldet die Station mit vorliegendem Sendewunsch diesen beim Arbitrator an. Dies geschieht im Rahmen der Verteilung des Wertes einer Variablen, der von der betroffenen Station erzeugt wird. Der Arbitrator reiht den Sendewunsch in Warteschlangen ein. Zur eindeutigen Identifikation der sendewilligen Station wird dabei der Bezeichner der Variablen gespeichert, aus deren Informationsrahmen die Kennung für die

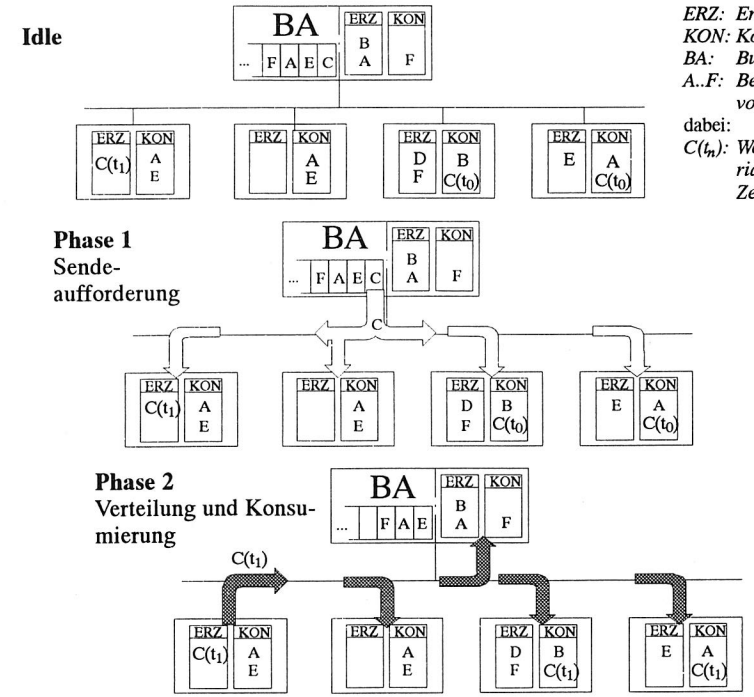


Bild 25: Variablenaustausch im Feldkommunikationssystem FIP

Anforderung entnommen wurde. Dieses Verfahren bevorzugt Stationen, die im Rahmen des zyklischen Nachrichtenaustausches häufig angesprochen werden.

Die Warteschlangen werden vom Arbitrator in den azyklischen Zeitfenstern abgearbeitet. Je nach Art des Übertragungswunsches, Variablen- bzw. Nachrichtenaustausch kommen dabei unterschiedliche Verfahren zur Anwendung.

Beim Austausch von Variablen wird die Station, die den Übertragungswunsch geäußert hat, aufgefordert, die Bezeichner der abzufragenden Variablen an den Arbitrator zu übermitteln. Dieser fragt – wiederum in einem azyklischen Zeitfenster – dann die bezeichneten Variablen ab. Das Vorgehen ist dabei analog zu dem beim zyklischen Austausch von Variablenwerten. Die Abarbeitung einer Anforderung für die Übertragung einer aperiodischen Nachricht beginnt ebenfalls mit dem Ansprechen der anfordernden Station. Kurzfristig wird vom Arbitrator die Buskontrolle an diese Station übergeben. Sie sendet stationsadressiert ihre Nachricht, wobei eine bestätigte oder unbestätigte Nachrichtenübertragung möglich ist. Die bestätigte Nachrichtenübertragung ist dabei nur auf einer Punkt-zu-Punkt Verbindung möglich. Das korrekte Eintreffen einer Nachricht wird von der empfangenden Station unverzüglich durch Senden einer Empfangsbestätigung angezeigt. Beim unbestätigten Nachrichtentransfer ist die Übertragung auch im Multicast möglich. Nach Abschluß der Nachrichtenübertragung wird die Buskontrolle jeweils vom Initiator des Nachrichtentransfers explizit an den Arbitrator zurückgegeben.

Eigenschaften der Anwendungsschicht

Die Anwendungsschicht des FIP-Feldkommunikationssystems besteht aus drei unabhängigen Blöcken: *ABAS*, einem nachrichtenorientierten Block mit den Protokollen *MCS* und *SubMMS* und MPS.

Das MCS Protokoll ist ein allgemeines Dienstelement der FIP-Anwendungsschicht. Es ist für die Kontrolle der Verbindung von Anwendungen sowie für die Übertragung der Daten zwischen den Anwendungen bei verbindungsorientierter und verbindungsloser Kommunikation zuständig und entspricht damit dem ACSE (*Association Control Service Element*)-Protokoll in der ISO-Kommunikationswelt. Findet eine verbindungslose Datenkommunikation statt, so enthalten die ausgetauschten Protokolldateneinheiten alle für eine korrekte Interpretation ihrer selbst notwendigen Daten [116]. Die angebotenen Dienste werden unter Verwendung der Dienste der Sicherungsschicht durch die Kooperation der Partnerinstanzen des MCS-Protokolls erbracht.

Das noch im Normierungsprozeß befindliche SubMMS-Protokoll [116] soll eine Untermenge der Dienste des MMS-Protokolls [35] sowie eine Untermenge der darin definierten Objekte enthalten. Die wesentlichen Konzepte von MMS, insbesondere das Client-Server-Prinzip und die Objektbasierung, sollen beibehalten werden. Neben den aus dem MMS-Standard übernommenen dynamisch auf- und abbaubaren Kommunikationsverbindungen soll SubMMS zusätzlich vordefinierte Verbindungen zwischen Applikationen sowie eine verbindungslose Kommunikation anbieten.

Als aus MMS zu übernehmende Objekte sind vorgesehen: das *Named Variable*-, das *Named Variable List*-, das *Semaphore*-, das *Event Condition*-, das *Event Action*-, das *Event Enrollment*-, das *Journal*-, das *Domain*- und das *Program Invocation*-Objekt. Die möglichen Attributwerte für diese Objekte werden gegenüber denen in der Definition der entsprechenden Objekte in MMS z.T. eingeschränkt sein.

Das in der Norm UTE NF C-46-602 ([117]) festgelegte MPS-Protokoll stellt dem Benutzer Dienste zum lokalen bzw. entfernten Zugriff auf Variablen bereit. Als Zugriffsfunktionen werden der Anwendung das Lesen des Wertes einer Variablen sowie dessen Schreiben angeboten. Des weiteren stehen Dienste zur Verfügung, mit deren Hilfe das Protokoll der Anwendungsschicht die Anwendung über den Empfang bzw. das Senden von Variablen in der Sicherungsschicht informieren kann.

Die Einbindung asynchroner Anwendungsprozesse in verteilte synchrone Anwendungen wird im MPS-Protokoll durch einen speziellen Puffermechanismus in der Anwendungsschicht ermöglicht. Zur Bewertung der Güte der Werte von ausgetauschten Variablen stehen die Evaluierung der „Promptness"- und der „Refreshment"-Statusinformation der Variablen zur Verfügung. Dabei wird über die Auswertung des „Refreshment"-Indikators in einer produzierenden Station festgestellt, ob der Wert einer Variablen rechtzeitig durch die Anwendung aktualisiert wurde. Über die Auswertung des „Promptness"-Status einer Variablen wird in einer konsumierenden Station bestimmt, ob der Wert einer Variablen rechtzeitig über das Netz ausgetauscht wurde. Sowohl für die Evaluierung des Refreshment- als auch des Promptness-Status existieren drei Bezugssysteme:

- *Asynchrones Bezugssystem*: Der Status der Variablen wird relativ zu ihrem Erzeugungszeitpunkt ausgewertet.
- *Synchrones Bezugssystem*: Der Status der Variablen wird relativ zu den Empfangszeitpunkten von *Synchronisationsvariablen* und der Erzeugungs- bzw. Verteilungsperiodizität ausgewertet.

- *Punktuelles Bezugssystem*: Der Status der Variablen wird ebenfalls relativ zu den Empfangszeitpunkten von *Synchronisationsvariablen* und der Erzeugungs- bzw. Verteilungsperiodizität ausgewertet. Dabei unterscheidet sich der Mechanismus dadurch von dem des synchronen Bezugssystems, daß während der anhaltenden Gültigkeit eines Variablenwertes erzeugte aktualisierte Werte berücksichtigt werden.

3.4.4 Bewertung von FIP

ANFORDERUNG EZ-1: Rechtzeitigkeit

FIP wurde als Echtzeit-Kommunikationssystem konzipiert und bietet Garantien bezüglich der rechtzeitigen Ausführung periodischer Dienste, insbesondere der periodischen Aktualisierung von in der verteilten Datenbasis gespeicherten Variablen. Dieser Mechanismus ist auf die Sicherungsschicht des Systems beschränkt. Die Anwendungen und die Mechanismen des MPS-Protokolls laufen in der Regel asynchron dazu. Aus ihrer Sicht ist der in der Sicherungsschicht nicht vorhandene Jitter maximal so groß, wie die Periodizität, mit der die Variable ausgetauscht wird. Die Konfiguration des periodischen Datenaustausches findet vor der Inbetriebnahme statt. Modifikationen, z.B. das Einplanen zusätzlicher Anforderungen, sind während des laufenden Betriebs nicht mehr möglich. Die beim aperiodischen Austausch von Variablen und Nachrichten auftretenden Verzögerungszeiten sind von einer Vielzahl von Einflußfaktoren abhängig. Es können hierfür keine oberen Zeitschranken angegeben werden.

ANFORDERUNG EZ-2: Berücksichtigung von Echtzeit- und Nicht-Echtzeit-Kommunikationsanforderungen

Das FIP System bietet vier verschiedene Datentransfermodi an. Durch die Erfüllung von Echtzeitanforderungen zeichnet sich dabei der periodische Austausch von Variablen und Nachrichten aus. Variable sind Kommunikationsobjekte und können periodisch oder aperiodisch ausgetauscht werden. Im Rahmen des periodischen Nachrichtenaustausches erhalten Stationen zu bestimmten Zeitpunkten die Möglichkeit, eine Nachricht zu senden. Ein aperiodisches Versenden von Nachrichten ist ebenfalls möglich. Die zu sendenden Nachrichten werden jeweils stationsintern in einer Warteschlange zwischengespeichert.

ANFORDERUNG EZ-3: Vorhersehbarkeit

Der Austausch von periodischen Variablen ist deterministisch und vorhersehbar. Eine Überlastung des Systems bzgl. des periodischen Variablenaustausches ist nicht möglich. Beim aperiodischen Nachrichtenaustausch kann es unter Überlast zu einem Überlauf der Warteschlange für anstehende Aufträge kommen [118].

Aperiodische Variablen- und Nachrichtentransfers sind bzgl. ihres Ausführungszeitpunktes nicht vorhersehbar. Das Verhalten einer Station im Fall von Überlast, d.h. wenn die Applikation mehr Anforderungen absetzt, als über das Netzwerk abgewickelt werden können, besteht in der Rückgabe einer entsprechenden Fehlermeldung.

ANFORDERUNG EZ-4: Zuverlässigkeit

Die Funktion des Busarbitrators ist für den Betrieb eines Systems entscheidend. Aus diesem Grunde ist die Bevorratung redundanter Busarbitratoren ebenso wie die redundanter Übertragungsmedien vorgesehen. Fällt eine Station aus, die als Produzent von Variablen fungiert, so sind Redundanzmechanismen nur auf Ebene der Anwendung möglich.

Eine Duplizierung von Daten kann nur bei Nachrichten auftreten. Die Erkennung duplizierter Nachrichten ist über die mitgeführte Sequenznummer (modulo 2^{15}) möglich. Eine Empfangskontrolle wird nur auf einer Punkt-zu-Punkt Nachrichtenverbindung durch das Protokoll unterstützt.

Die Gleichberechtigung aller Stationen im System erlaubt den spontanen Datenaustausch zur Information über aufgetretene Fehler etc. zwischen beliebigen Stationen zu beliebigen Zeitpunkten.

ANFORDERUNG ***EZ-5: Verarbeitung von Echtzeit-Prozeßvariablen***

FIP bietet keine Unterstützung für die Verarbeitung von zeitgestempelten Daten. Die Darstellung einer zeitkritischen Variable ist durch Verwendung von strukturierten Variablen möglich. Der Rückgriff auf ASN.1 [119] erlaubt die Verwendung des Standarddatentypen *Generalised Time*. Zudem ist die Verwendung des Datentyps *Binary Time* möglich. Die maximale Zeitauflösung beträgt im ersten Fall eine Sekunde, im letzteren 10 µs.

Zur Überprüfung der Konsistenz von Daten werden die Auswertung des Refreshment- und des Promptness-Status angeboten. Daneben besteht die Möglichkeit der Evaluierung der Identität der verteilten Kopien für Listen von Variablen (*Spatial Consistency*). Ein Attribut zur Anzeige der entfernten Gültigkeit eines Datenwertes ist nicht vorgesehen.

ANFORDERUNG ***EZ-6:*** **Unterstützung einer verteilten Uhr**

Eine verteilte Uhr wird durch FIP nicht unterstützt. Überlegungen und ein Algorithmus, um diese in Software auf Basis der angebotenen Dienste zu realisieren, sind in [120] dargestellt.

ANFORDERUNG ***VS-1: Benutzergesteuerte Prioritätenvergabe***

Eine explizit benutzergesteuerte Prioritätenvergabe gibt es in FIP nur bei den aperiodischen Diensten des MPS-Protokolls. Hier kann der Benutzer beim Dienstaufruf zwischen zwei Prioritätsstufen wählen, wobei in beiden Prioritätsstufen keine Echtzeitfähigkeit angeboten wird. Die Auswahl einer Prioritätsstufe wirkt sich auf die Reihenfolge der Bearbeitung der aperiodischen Anforderung durch den Busarbitrator aus.

ANFORDERUNG ***VS-2: Synchronisation von Applikationen***

Das MPS-Protokoll bietet Mechanismen zur Synchronisation a priori asynchron laufender Prozesse an. Hier findet ein Mechanismus Anwendung, der es erlaubt, aktualisierte Variablenwerte im gesamten Netzwerk gleichzeitig „freizuschalten". Zu diesem Zweck sind *Synchronisationsvariable* vorgesehen.

ANFORDERUNG ***VS-3: Scheduling des Kommunikationssystems***

Die Zuordnung des Zugriffs auf das gemeinsam genutzte Kommunikationsmedium erfolgt vollständig unter der Kontrolle des Busarbitrators anhand der dort vorhandenen Tabellen für den periodischen und der gespeicherten Anforderungen für den aperiodischen Datenaustausch. Das Scheduling des Kommunikationssystems für periodische Datenübertragung erfolgt während der Konfigurierung. Eine dynamische Änderung der Tabellen wird nicht unterstützt.

ANFORDERUNG ***VS-4: Breites Spektrum an PDU-Sequenzen***

FIP unterstützt alle vier geforderten Typen von PDU-Sequenzen. Typ 1 und Typ 2 werden durch den unbestätigten bzw. bestätigten Nachrichtenaustausch zwischen genau zwei Stationen realisiert. Die PDU-Sequenz vom Typ 3 wird durch das Senden von Nachrichten an Stationsgruppen repräsentiert. Der Austausch von Variablenwerten basiert auf der PDU-Sequenz vom Typ 4.

ANFORDERUNG VS-5: Autonomie

Die Verwaltung und Steuerung von Applikationen in einer Station wird nur durch die vorgesehenen Mechanismen des SubMMS-Protokolls unterstützt.

ANFORDERUNG VS-6: Komfortable Kommunikationsdienste

Das MPS-Protokoll definiert eine Reihe von Kommunikationsdiensten, die es erlauben, auf definierte Variable zuzugreifen und Nachrichten zu senden. Spezielle Dienste zum Zugriff auf Statusinformationen der Stationen oder zum Laden und Starten von Programmen sind dort nicht vorgesehen. Als funktionale Untermenge des MMS-Standards ist neben dem MPS-Protokoll das SubMMS-Protokoll vorgesehen, das entsprechende Mechanismen beinhalten soll.

ANFORDERUNG VS-7: Berücksichtigung von ereignis- und zustandsgesteuerter Kommunikation

FIP unterstützt sowohl ereignis- als auch zustandsgesteuerte Kommunikation. Die zustandsgesteuerte Kommunikation ist tabellengesteuert und beinhaltet den periodischen Variablen- und Nachrichtenaustausch. Die ereignisgesteuerte Kommunikation wird durch die aperiodischen Kommunikationsmechanismen von FIP repräsentiert. Die konzeptionelle Auslegung des Systems legt dabei eine eindeutige Ausrichtung auf die zustandsgesteuerte Kommunikation fest.

ANFORDERUNG VS-8: Unterstützung von Replikationsmechanismen für Daten

FIP basiert für den Austausch von Werten von Variablen auf dem Prinzip der verteilten Datenbasis. Dabei liegen die von einer Station erzeugten Daten als lokale Kopien in allen Verbraucherstationen vor. Die unterstützten Datenkonsistenzattribute sind bereits unter den Ausführungen zu Anforderung 11 dargestellt.

ANFORDERUNG MM-1: Übertragung kontinuierlicher und diskreter Medien

FIP eignet sich prinzipiell sowohl für die Übertragung kontinuierlicher als auch diskreter Medien. Der isochrone Transport von Daten, die kontinuierliche Medien repräsentieren, kann mittels der periodischen Datentransferdienste durchgeführt werden. Die Übertragung diskreter Medien kann sowohl mit periodischen als auch mit aperiodischen Datentransferdiensten oder mittels der Dienste und Objekte des SubMMS-Protokolls realisiert werden. Die Verwendung von Mechanismen des periodischen Datenaustausches bedeutet jedoch bei nicht vorliegendem Übertragungsbedürfnis eine Verschwendung von Übertragungsbandbreite. Das MPS-Protokoll unterstützt dabei keine Segmentierung von größeren Datenblöcken. Alle unbestätigten Datentransfermechanismen unterstützen die Übertragung von Informationen von einer Quellstation an Stationsgruppen.

Während der Laufzeit des Systems kann isochroner Datentransfer nur dann durchgeführt werden, wenn er bereits während der Konfigurationsphase des Systems berücksichtigt wurde. Eine dynamische Ein- oder Ausplanung ist nicht möglich.

In der Quellstation werden isochron zu übertragende Daten maximal um die Dauer der Periodizität der benutzten Variable verzögert. Bei der Verwendung von periodischen Nachrichten ist die Verzögerungszeit abhängig von der Anzahl bereits eingelasteter periodischer Nachrichtenübertragungsaufträge. In den Zielstationen besteht die Möglichkeit, die Applikation über das Eintreffen eines neuen Wertes zu informieren. Diese ist aber gefordert, die Daten selbst aus dem lokalen Speicher auszulesen.

ANFORDERUNG MM-2: Unterstützung von Aufnahmesystemen durch das Kommunikationsprotokoll

Das MPS-Protokoll von FIP bietet keine Unterstützung für die Steuerung von Geräten. Das SubMMS-Protokoll basiert auf dem Konzept des virtuellen Fertigungsgerätes, so daß diesbezüglich die gleichen Randbedingungen gelten, wie für das PROFIBUS System.

ANFORDERUNG MM-3: Synchronisation von Multimedia-Datenströmen

FIP unterstützt die Synchronisation vom multimedialen Datenströmen implizit durch den Mechanismus zum periodischen Datenaustausch. Dieser gewährleistet, daß Dateneinheiten in einer definierten Reihenfolge periodisch ausgetauscht werden.

3.5 Zusammenfassung

Es zeigt sich, daß konzeptbedingt keines der beiden Systeme alle eingangs aufgestellten Anforderungen erfüllt, wobei beide Systeme in unterschiedlichen Bereichen Stärken aufweisen. FIP bietet dabei insbesondere eine gute Unterstützung für die zustandsgesteuerte Echtzeitkommunikation, wohingegen PROFIBUS in größerem Maße den Anforderungen der ereignisgesteuerten Kommunikation und der Autonomie der Stationen entsprechen kann. Eine zusammenfassender Überblick über die Bewertung ist in Bild 26 dargestellt. Der Grad der Erfüllung einer Anforderung durch ein Feldkommunikationssystem wird dabei durch durch die Lage der zugeordneten Linie auf der Anforderungsachse repräsentiert.

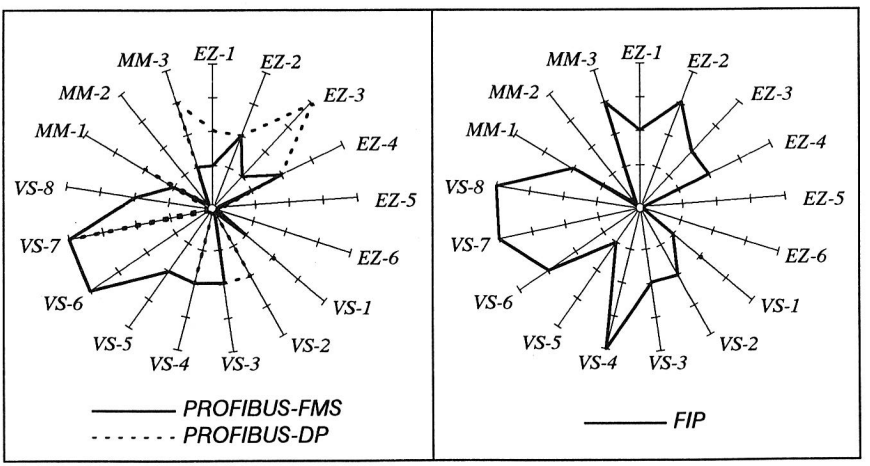


Bild 26: Vergleichende Bewertung von PROFIBUS und FIP

4 Erweiterung und Nutzung vorhandener Systeme

Ein möglicher Weg zur Erfüllung der Anforderungen besteht in der Erweiterung bzw. in der angepaßten Nutzung vorhandener Systeme. Nachfolgend werden die Ergebnisse der Ansätze,

- FIP um die Integration einer verteilten Uhr mit den zugehörigen Protokollmechanismen zu erweitern und
- basierend auf PROFIBUS Multimedia-Daten zu übertragen, vorgestellt.

Der Schwerpunkt der Darstellungen liegt dabei auf ersterem Ansatz, der eine Erweiterung eines Systems im Gegensatz zu einer angepaßten Nutzung im letzteren Fall darstellt.

4.1 OLCHFA: Ein zeitkritisches Feldkommunikationssystem

Auf Basis des FIP-Systems wurde im OLCHFA (*An Open, Low-Cost Time Critical Wireless Fieldbus Architecture*)-Projekt[2] versucht, ein vorhandenes Feldkommunikationssystem um die benötigten Funktionen zur Behandlung von zeitbehafteten Daten zu erweitern [121, 122]. Für die hier betrachtete Erweiterung des Spektrums an der Anwendungsschnittstelle angebotener Dienste stehen neben den Funktionen des MPS-Protokolls von FIP zusätzlich Dienste zum Zugriff auf eine

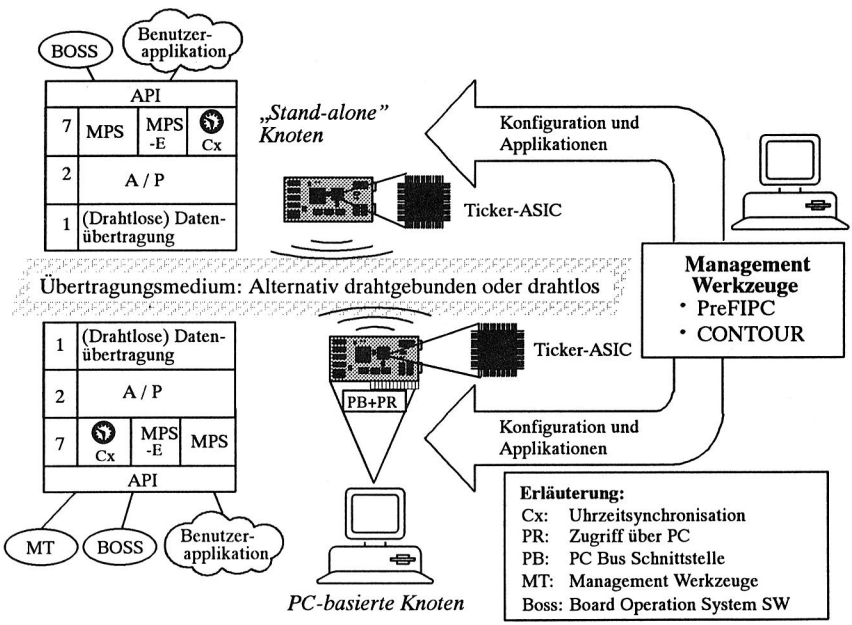


Bild 27: Übersicht über das OLCHFA-Gesamtsystem

systemweit synchronisierte Uhrzeit zur Verfügung. Des weiteren werden die entwickelten Werkzeuge zur Unterstützung der Konfigurierung des Systems vorgestellt. Der Gesamtumfang des OLCHFA-Projektes ist in Bild 27 dargestellt.

2. Förderung durch Kommission der EG im ESPRIT-Programm (EP 7210).

4.1.1 MPS-E: Ein Überblick

Das MPS-Protokoll des Kommunikationssystems FIP verfügt im wesentlichen über vier Produktivdienste für den Austausch von Werten von Variablen (vgl. [117]). Diese Dienste wurden gemäß der Anforderungen nach einer Echtzeitdatenverarbeitung mit expliziter Zeitbehandlung erweitert. Der Forderung nach Berücksichtigung von *zustands-* und *ereignisgesteuerter* Kommunikation wird durch die Konzeption und Implementierung der Ereignisbehandlung Rechnung getragen. Die Erweiterung des Dienstespektrums führte zudem zu der Einführung neuer Datentypen. Die erweiterte Version des MPS-Protokolls wird mit *MPS-E* (MPS-Extended) bezeichnet [123]. Die Einordnung des MPS-E Protokolls in die Kommunikationsarchitektur des OLCHFA-Systems ist in Bild 28 dargestellt.

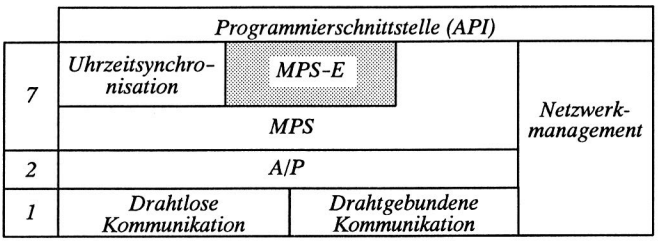


Bild 28: Kommunikationsarchitektur des OLCHFA-Systems

Überblick über das erweiterte Dienstespektrum

Die durch FIP im MPS-Protokoll bereitgestellten Basisdienste [117, 124] erlauben es den Applikationen, den Wert einer produzierten Variable nur in die lokale oder zusätzlich auch in die entfernten Instanzen der verteilten Datenbasis zu schreiben. Konsumierende Applikationen können sowohl lokal als auch entfernt lesend auf den Wert von Variablen zugreifen.

Von einem Echtzeitsystem mit expliziter Zeitbehandlung wird gefordert, daß Werte von Echtzeitvariablen nur dann in das System eingebracht werden können, wenn sie noch gültig sind. Zur Überprüfung der Gültigkeit werden dabei die vorkonfigurierte *Gültigkeitszeitdauer* und die zur Laufzeit bestimmte *Erzeugungszeit* herangezogen. Des weiteren wird gefordert, daß Werte von Echtzeitvariablen nur dann von einer Applikation gelesen werden können, wenn sie zum Zeitpunkt des Zugriffs in Bezug auf ihre Gültigkeitszeitdauer und ihre Erzeugungszeit noch gültig sind. Diese Konzepte sind dabei sowohl auf periodisch als auch auf aperiodisch ausgetauschte Variable anwendbar. Neben den dafür benötigten Diensten ist in MPS-E ein lokaler Dienst definiert, der die Applikation von dem Auslaufen der Gültigkeitszeitdauer des aktuellen Wertes einer produzierten Variablen in Kenntnis setzt.

Die Möglichkeit, im OLCHFA-Feldkommunikationssystem auf eine netzweit synchronisierte Uhrzeit zugreifen zu können, bietet zusammen mit dem Datentyp der Echtzeitvariablen und den erweiterten Diensten der Anwendungsschicht zudem auf semantischer Ebene eine Alternative zu den Auswertungsmechanismen des Refreshment- und Promptness-Status an, die in den derzeitigen Implementierungen von FIP nur unvollständig realisiert sind [125].

Erweiterte Spezifikation der Variablen

Die im MPS-Standard ([117]) angeführte generische Spezifikation für die Klasse der Variablen des MPS-Protokolls der FIP-Anwendungsschicht wurde gemäß der Anforderungen der zeitkritischen Protokollerweiterungen ergänzt. Zu diesem Zweck wurden zwei neue Klassen von Variablen eingeführt: Die *Echtzeitvariable* und die *Ereignisvariable*. Von dieser Erweiterung sind sowohl die Klassenattribute als auch die Instanzattribute der zugrundeliegenden Definition aus FIP betroffen. Für eine detaillierte Darstellung der sich ergebenden neuen Klassendefinitionen sei auf die diesbezüglichen Darstellungen in [123] verwiesen.

Echtzeitvariable

Der Aufbau einer Echtzeitvariablen ist in Bild 29 in der abstrakten Syntaxnotation ASN.1 dargestellt. Die zur Verfügung stehenden Alternativen für den Typ der Prozeßwerte und deren in Kommentarform angegebene Größe basieren auf den entsprechenden Definitionen von MPS.

```
TC-VariablePDU   ::= SEQUENCE {     -- maximale Länge:128 Byte
    var-id           A-KEY,         -- Identifikation
    creation-time    TIMESTAMP,     -- Erzeugungszeitpunkt
    validity-period  PERIOD,        -- Gültigkeitszeitraum
    data-consistency BIT STRING     -- Konsistenzattribute
    {
        remote-validity  (0),       -- technische Gültigkeit
        merrit-ok        (1)        -- Synchronisationsstatus
    },
    value    SEQUENCE OF CHOICE     -- (Folge von) Prozeßwert(en)
    {
        BOOLEAN,                    -- Boolesche Variable
        Unsigned8,                  -- 8 Bit Ganzzahl ∈ N0
        Unsigned16,                 -- 16 Bit Ganzzahl ∈ N0
        Unsigned32,                 -- 32 Bit Ganzzahl ∈ N0
        Integer8,                   -- 8 Bit Ganzzahl
        Integer16,                  -- 16 Bit Ganzzahl
        Integer32,                  -- 32 Bit Ganzzahl
        Sfpoint,                    -- 32 Bit Gleitkommazahl
        Dfpoint,                    -- 64 Bit Gleitkommazahl
        BIT STRING,                 -- Folge einzelner Bits
        OCTET STRING,               -- Folge beliebiger Zeichen
        VisibleString               -- Folge lesbarer Zeichen
    }
}
```

Bild 29: Definition einer Echtzeitvariable in ASN.1

Die Änderungen der vorkonfigurierten Gültigkeitszeitdauer des Wertes einer Variablen ist in der produzierenden Station auch im laufenden Betrieb möglich. Dies ist z.B. dann hilfreich, wenn die Werte einer Variable während des Anlaufs eines technischen Prozesses häufig, während der Betriebsphase jedoch nur selten ausgetauscht werden müssen und damit aus Sicht der Applikation auch eine variierende Gültigkeitszeitdauer haben. Die Übertragung und Speicherung der strukturierten Variablen erfolgt in einer geeigneten Kodierung.

Ereignisvariable

Die Ereignisvariable wird zur Übermittlung von Benachrichtigungen über den Eintritt eines oder mehrerer Ereignisse genutzt. Ihr struktureller Aufbau in ASN.1 ist in Bild 30 dargestellt. Die Er-

```
EV-VariablePDU   ::= SEQUENCE {
    var-id           A-KEY,        -- Identifikation
    occurrence-time  TIMESTAMP,    -- Erzeugungszeitpunkt der
                                   -- verursachenden Variablen
    mask             BIT STRING    -- Ereignisidentifikation
                                   -- (benutzerdefiniert)
}
```

Bild 30: Definition einer Ereignisvariable in ASN.1

eignisvariable erlaubt die Verschlüsselung von bis zu 32 verschiedenen Ereignissen pro Station, die in beliebigen Kombinationen auftreten können. In dem zur Variable gehörigen Zeitstempel wird der Zeitpunkt der Feststellung des Ereignisses hinterlegt. Dieser entspricht dem Erzeugungszeitpunkt des Wertes der beobachteten Variablen, die das Ereignis ausgelöst hat. Für die Übertragung wird die strukturiert aufgebaute Ereignisvariable kodiert.

4.1.2 Lokale Dienste

Die in MPS-E konzipierten und implementierten Erweiterungen betreffen sowohl lokal als auch entfernt wirkende Dienste. Die Konventionen bezüglich der in der gesamten Arbeit genutzten Darstellungsformen für die Parameter der Kommunikationsdienste und die Dynamik des Nachrichtenaustausches sind in Anhang A zusammengefaßt.

Zeitkritisches lokales Schreiben einer Variablen

Als Pendant zum Dienst für das lokale Schreiben einer Variablen ist der zeitkritische lokale Dienst `A-WRITELOC-TC` konzipiert. Für diesen Dienst sind die Dienstelemente der Anforderung und der Bestätigung definiert. Die Dienstbestätigung erfolgt lokal und enthält Informationen über das Resultat der Ausführung des Dienstes. Den Dienstelementen sind dabei als Argument bzw. Rückgabewert die in Tabelle 4 aufgeführten Parameter zugeordnet:

Parametername	req	cnf
Argument	**M**	
Variable specification	M	
Value	M	
Remote validity	M	
Creation time	M	
Validity period	M	
Result(+)		**S**
Result(–)		**S**
Error specification		M

Tabelle 4: Parameter der zeitkritischen Schreibdienste

Die Identifikation der Variablen ist im Parameter „Variable specification", der zu schreibende Wert im Parameter „Value" enthalten. Der Wert des Parameters „Remote validity" zeigt die Gültigkeit des Wertes aus technischer Sicht an. Der Erzeugungszeitpunkt der Variablen wird im Parameter „Creation Time" übergeben. Im OLCHFA-System beträgt die Granularität des Zeitstempels

dabei 1 µs. Die Dauer der Gültigkeit ab dem Erzeugungszeitpunkt wird durch den Wert des Parameters „Validity period" als ganzzahliges Vielfaches von 32 µs angezeigt. Für diesen Parameter sind zwei ausgezeichnete Werte zur Anzeige der Verwendung der während der Konfigurierung festgelegten Gültigkeitsdauer und zur Anzeige einer zeitlich nicht eingeschränkten Gültigkeit definiert. Über die durch das FIP-System festgelegten Fehlermeldungen für die Dienstbestätigung hinaus sind solche definiert, die sich auf die zeitliche Gültigkeit der Variablen beziehen.

Die Instanz von MPS-E überprüft nach Aufruf des lokalen Schreibdienstes die zeitliche Gültigkeit der Variablen. Ist diese nicht gegeben, so wird die Schreibanforderung nicht ausgeführt und es wird statt dessen eine entsprechende Fehlermeldung an den Dienstbenutzer übergeben. Anderenfalls wird der lokale Schreibdienst von MPS dazu genutzt, die vorher kodierte Variable in der lokalen Instanz der verteilten Datenbasis zu aktualisieren. Falls diese Variable für die Überwachung durch die Ereignisbehandlung (vgl. Kap. 4.1.4) konfiguriert wurde, wird von dieser überprüft, ob der aktuelle Wert eine oder mehrere Ereignisbedingung erfüllt.

Zeitkritisches lokales Lesen einer Variablen

Für das zeitkritische lokale Lesen sind die Dienstelemente der Dienstanforderung und -bestätigung mit den zugehörigen, in Tabelle 5 aufgeführten, Parametern definiert. Neben dem Wert der Variablen werden an der Benutzerschnittstelle auch die der Variablen zugeordneten Konsistenzinformationen bereitgestellt.

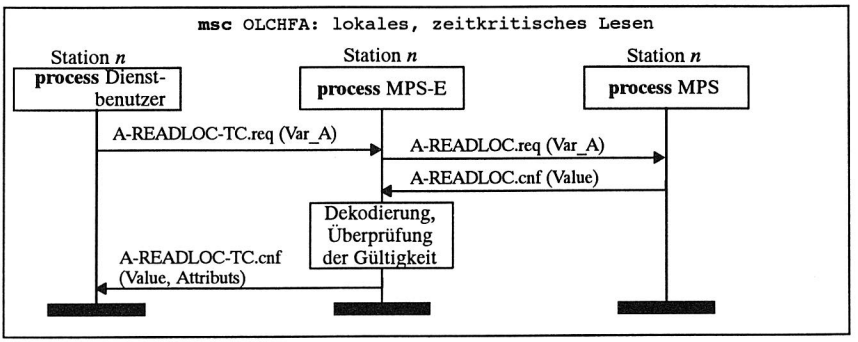
Die Bedeutung der Parameter ist teilweise bereits in den obigen Ausführungen enthalten und wird hier nur um die der zusätzlichen Parameter ergänzt. In der Dienstanforderung besteht für den Benutzer die Möglichkeit, durch Setzen des Parameters „Forced" das Lesen eines Wertes unabhängig von der Auswertung der besonderen Attribute der Echtzeitvariable durchzuführen. Die durch das FIP-Basissystem bereitgestellten Konsistenzinformationen der rechtzeitigen Aktualisierung des Wertes in der produzierenden Station, sowie dessen rechtzeitige Aktualisierung in der verteilten Datenbasis werden in den Parametern „Refreshment status" und „Promptness status" bereitgestellt. Die Menge der durch FIP gegebenen Fehlerspezifikationen („Error specification") ist um solche zur Anzeige zeitlicher Ungültigkeit der Werte ergänzt worden.

Parametername	req	cnf
Argument	**M**	
Variable specification	M	
Forced	M	
Result(+)		**S**
Value		M
Remote validity		M
Creation Time		M
Validity period		M
Refreshment status		C
Promptness status		C
Result(-)		**S**
Error specification		M

Tabelle 5: Parameter der zeitkritischen Lesedienste

Nach Entgegennahme der Dienstanforderung liest die MPS-E Instanz den kodierten Wert der Variablen mittels des lokalen Lesedienstes von FIP aus, dekodiert ihn und überprüft die zeitliche Gültigkeit der Variablen. Ist diese gegeben, so wird der Wert der Variablen samt der zugehörigen

Werte der Konsistenzattribute an den Dienstbenutzer übergeben; anderenfalls wird eine negative Dienstbestätigung unter Angabe der Fehlerspezifikation abgesetzt. Die Abfolge der Dienstaufrufe bei fehlerfreier Dienstausführung ist in Bild 31 dargestellt.



A-READLOC-TC.req(Var_A) : *Dienstanforderung: Zeitkritisches lok. Lesen der Variable „Var_A"*

A-READLOC.req(Var_A): *Dienstanforderung: Lokales Lesen der Variable „Var_A"*

A-READLOC.cnf(Value): *Dienstbestätigung: Wert der Variable „Var_A"*

A-READLOC-TC.cnf(Value, Attributs): *Dienstbestätigung: Wert und zeitbezogene Attribute der Variablen „Var_A"*

Bild 31: Zeitkritisches lokales Lesen einer Echtzeitvariable

Ungültigkeitsanzeige einer Echtzeitvariablen

Der Ablauf der Gültigkeit einer Echtzeitvariable kann – bei entsprechender Konfigurierung der Variablen – der Applikation der produzierenden Station durch den Dienst `A-INVALID` angezeigt werden. Als Dienstelement ist für diesen Dienst nur die lokale Dienstankündigung definiert, die als einzigen Parameter die Identifikation der Variablen enthält, deren Gültigkeitszeitdauer abgelaufen ist. Die zeitliche Abfolge der Dienste bei der lokalen Ungültigkeitsanzeige ist in Bild 32 zusammen mit dem Vorgang des lokalen Schreibens einer Echtzeitvariable dargestellt.

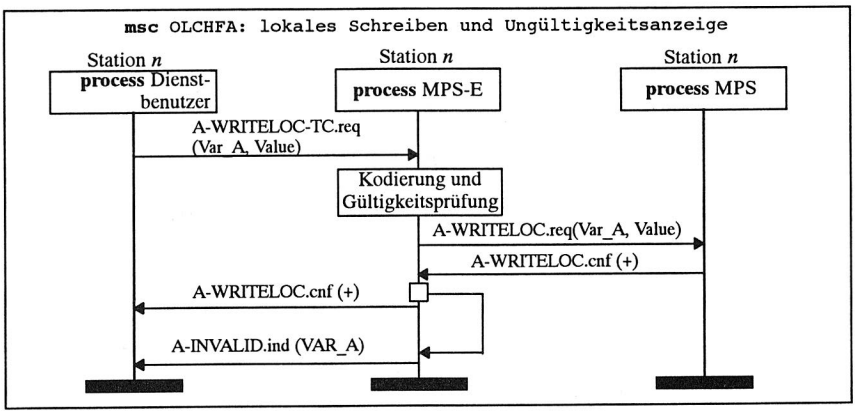
4.1.3 Entfernte Dienste

Neben den zusätzlichen lokalen Diensten sind zeitkritische Dienste definiert, die bei einer oder mehreren entfernten OLCHFA-Stationen wirken. Sie dienen dem Auslesen des Wertes einer Echtzeitvariable aus dem Speicher der produzierenden Station bzw. dem Aktualisieren einer lokal produzierten Variable in den Speichern der konsumierenden Stationen. Alle entfernt wirkenden zeitkritischen Dienste werden grundsätzlich mit der hohen Priorität des FIP-Systems abgewickelt.

Zeitkritisches entferntes Schreiben einer Variablen

Dieser Dienst beinhaltet die notwendigen Mechanismen, um den lokal erzeugten Wert einer Variablen explizit in die Pufferspeicher der als Konsumenten dieser Variablen deklarierten Stationen zu schreiben. Wie auch beim zeitkritischen lokalen Schreiben, so wird ein Wert nur dann übernommen, wenn seine temporale Gültigkeit noch gegeben ist. Ist dies nicht der Fall, so erfolgt die Übergabe einer entsprechenden Fehlermeldung an den Dienstbenutzer. Den Dienstelementen sind die in Tabelle 4 aufgeführten Parameter zugeordnet.

Falls die entfernt zu schreibende Variable auf die Erfüllung einer Ereignisbedingung hin beobachtet wird, so wird der übergebene Wert zudem mit dem oder den spezifizierten Grenzwerten vergli-



A-WRITELOC-TC.req(Var_A, Value): *Dienstanforderung: Zeitkritisches lokales Schreiben der Variable „Var_A" mit dem Wert „Value"*

A-WRITELOC.req(Var_A, Value): *Dienstanforderung: Lokales Schreiben der Variable „Var_A" mit dem Wert „Value"*

A-WRITELOC.cnf(+): *Positive Dienstbestätigung*

A-WRITELOC-TC.cnf(+): *Positive Dienstbestätigung (zeitkritisches Schreiben)*

A-INVALID.ind(Var_A): *Dienstankündigung: Gültigkeit der Variablen „Var_A" abgelaufen.*

Bild 32: Zeitkritisches lokales Schreiben und Ungültigkeitsanzeige einer Variablen

chen (vgl. Kap. 4.1.4), bevor er kodiert wird. Die zeitliche Abfolge der Dienste und deren Abbildung auf die Dienste des MPS-Protokolls beim zeitkritischen entfernten Schreiben wird durch Bild 33 verdeutlicht.

Zeitkritisches entferntes Lesen einer Variablen

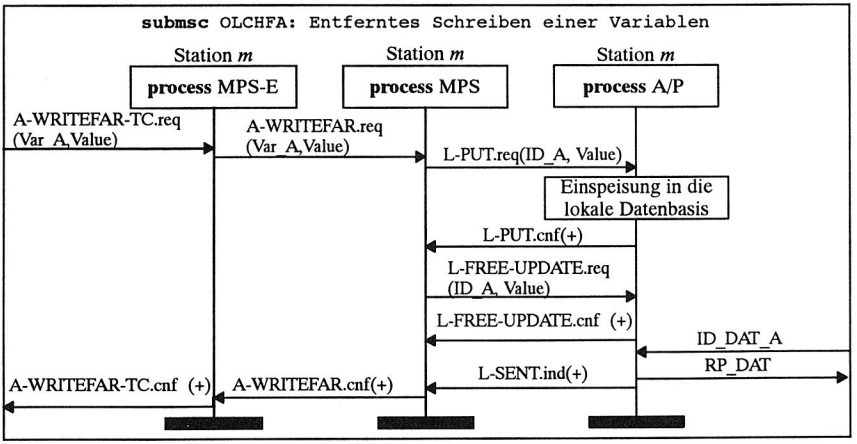
Mit Hilfe dieses Dienstes kann der Konsument einer Echtzeitvariable explizit den aperiodischen Austausch einer durch eine andere Station produzierten Variablen anstoßen, bevor auf die lokale Kopie von deren Wert zugegriffen wird. Als Dienstelemente sind für diesen Dienst die Anforderung und die Dienstbestätigung definiert. Die Parameter der Dienstelemente entsprechen denen des zeitkritischen lokalen Lesedienstes (vgl. Tabelle 5). Wie auch der zeitkritische entfernt wirkende Schreibdienst, so wird auch dieser Dienst grundsätzlich mit der hohen Priorität des FIP-Systems ausgeführt. Die Überprüfung der Gültigkeit des Wertes einer Variablen vor der Weitergabe an die Applikation entspricht der, die auch für lokal gelesene Variable vorgenommen wird.

4.1.4 Ereignisbehandlung

Bei der Ereignisbehandlung handelt es sich um einen für das OLCHFA- System neu konzipierten Mechanismus. Ziel ist es, die Überprüfung, ob Ereignisse im technischen Prozeß aufgetreten sind, sowie den Anstoß von Funktionen zur deren Behandlung durch das Kommunikationssystem zu unterstützen.

Funktionsweise

Die Ereignisbehandlung von MPS-E überprüft beim lokalen oder entfernten zeitkritischen Schreiben des Wertes einer Variablen, ob eine Ereignisbedingung erfüllt ist. Liegt eine derartige Situa-



A-WRITEFAR-TC.req(Var_A, Value):	*Dienstanforderung: Zeitkritisches entferntes Schreiben der Variable „Var_A" mit dem Wert „Value"*
A-WRITEFAR.req(Var_A, Value):	*Dienstanforderung: Entferntes Schreiben der Variable „Var_A" mit dem Wert „Value"*
L-PUT.req(ID_A, Value):	*Lokales Schreiben der Variablen A*
L-PUT.cnf(+):	*Dienstbestätigung für das lokale Schreiben*
L-FREE-UPDATE.req(ID_A, Value):	*Anforderung für aperiodischen Variablenaustausch*
L-FREE-UPDATE.cnf(+):	*Bestätigung für Übernahme der Dienstanforderung*
ID_DAT_A:	*Anforderung für Austausch der Variablen „A"*
RP_DAT_A	*Austausch des Wertes der Variablen „A"*
L-SENT.ind(+):	*Lokale Dienstankündigung: Senden ist erfolgt*
A-WRITEFAR.cnf(+):	*Positive Dienstbestätigung*
A-WRITEFAR-TC.cnf(+):	*Positive Dienstbestätigung (zeitkritisches Schreiben)*

Bild 33: Zeitkritisches entferntes Schreiben einer Variablen

tion vor, so wird eine Benachrichtigung über den Eintritt des Ereignisses gesendet. Ereignisse sind dabei als Über- bzw. Unterschreitung von vorgegebenen Grenzwerten durch Werte von Echtzeitvariablen definiert. Die Kennung des eingetretenen Ereignisses wird in dem Wert einer ausgezeichneten Variable, der *Ereignisvariable*, verschlüsselt. Pro Station ist genau eine, durch ihren Namen eindeutig bestimmte, Ereignisvariable definiert. Ihr Name wird durch Konkatenation des Präfixes „EV_" und des eindeutigen Namens der Station gebildet. Diese Variable erlaubt die eindeutige Kodierung von bis zu 32 gleichzeitig auftretenden Ereignissen pro Station, die beliebig den dort produzierten Echtzeitvariablen zugewiesen werden können.

Die Festlegung der Ereignisbedingungen *(Event Conditions)*, die Bestimmung der Stationen die an der Behandlung der einzelnen Ereignisse beteiligt sind *(Event Enrollment)* und die Zuordnung der dort nach Eintritt eines Ereignisses auszuführenden Aktionen *(Event Actions)* erfolgt während der Konfigurierung des Systems (vgl. Bild 34). Für jedes Ereignis lassen sich $m \in [0..k\text{-}1]$ Stationen festlegen, die an der Behandlung des Ereignisses beteiligt sind, wenn k die Anzahl der Stationen im Netzwerk ist.

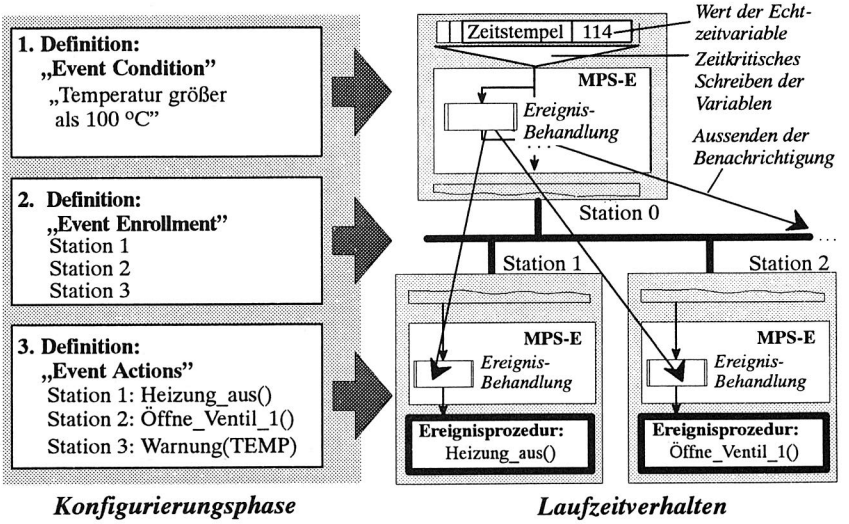


Bild 34: Ereignisbehandlung: Konfigurierung und Laufzeitverhalten

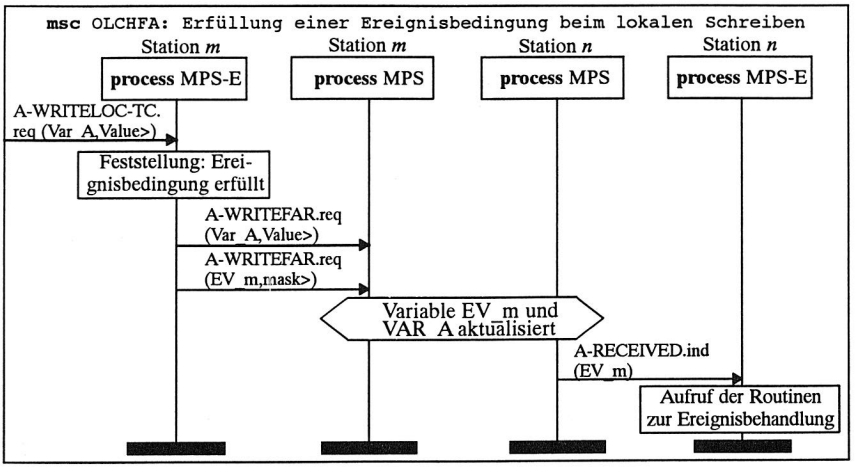
Bei der Festlegung der Ereignisbedingungen werden drei verschiedene Typen von Ereignissen unterschieden:

- *Ascending Threshold Event*: Der aktuelle Wert einer Echtzeitvariablen ist größer als der für diese Variable spezifizierte Grenzwert.
- *Descending Threshold Event*: Der aktuelle Wert einer Echtzeitvariablen ist kleiner als der für diese Variable spezifizierte Grenzwert.
- *Deadband Event*: Der aktuelle Wert einer Echtzeitvariablen ist entweder größer oder kleiner als der für diese Variable spezifizierte obere bzw. untere Grenzwert.

Wird die Erfüllung einer Ereignisbedingung bei einem zeitkritischen lokalen Schreiben einer Variable festgestellt, so wird vor dem Senden der Mitteilung über den Eintritt eines Ereignisses der aperiodische Austausch der verursachenden Variablen vorgenommen, um die verteilten Kopien der Variable zu aktualisieren. Dieser Vorgang wird durch die Darstellungen in Bild 35 verdeutlicht.

Die gesendete Ereignisvariable enthält einen Zeitstempel, der den Erzeugungszeitpunkt des Wertes enthält, der die Ereignisbedingung erfüllt hat. Damit verfügen die an der Ereignisbehandlung beteiligten Stationen über den genauen Zeitpunkt des Eintritts des Ereignisses und können u.a. durch Vergleich der Erzeugungszeitpunkte feststellen, ob der lokal verfügbare Wert der Variablen der ist, der die Ereignisbedingung erfüllt hat.

Mit Hilfe des *Deadband Event* können beispielsweise *Zweipunkt-Regelungen* sehr einfach realisiert werden, was am Beispiel einer Temperaturregelung, z.B. eines Kessels, verdeutlicht sei: Die Temperatur des Kesselinhaltes, der ständig Wärme an die Umgebung abgibt, soll in einem festgelegten Bereich gehalten werden. Fällt sie unter einen vorgegebenen Grenzwert, so wird nach dem Aussenden der Benachrichtigung autonom durch die Ereignisbehandlung der OLCHFA-Station,



A-RECEIVED.ind(EV_M): Dienstankündigung: Variable „EV_m" wurde über aktualisiert.

Bild 35: Senden einer Ereignismitteilung im Zusammenhang mit einem lokal geschriebenen Wert einer Variablen

die zur Heizung gehört, diese eingeschaltet. Sie wird nach Eintreffen einer weiteren Alarmmeldung, die bei Überschreitung des oberen Grenzwertes gesendet wird, wieder ausgeschaltet. Eine Überwachung der Temperatur und der Aufruf der notwendigen Routinen zur Ereignisbehandlung wird durch MPS-E autonom vorgenommen.

4.1.5 Konfigurierung des OLCHFA-Systems

Der Wechsel von einer herkömmlichen, analogen Verdrahtungen auf Feldkommunikationssysteme geht bzgl. der Systemkonfiguration mit dem Schritt von einer vergleichsweise einfach vorzunehmenden Anbindung einer derartigen Schnittstelle zu einer zum Teil sehr komplexen Parameterisierung und Handhabung eines Kommunikationsprotokolls einher. Daraus resultiert die Forderung nach einer möglichst weitgehenden Anwenderunterstützung für diese Tätigkeit, die mit zunehmender Komplexität der Systeme an Bedeutung gewinnt. Für viele Feldkommunikationssysteme, z.B. PROFIBUS oder INTERBUS-S, existieren derartige Werkzeuge [126, 127].

Gegenüber der Konfigurierung von Netzen in den höheren Ebenen der rechnerintegrierten Fertigung, z.B. LANs, sind bei Feldkommunikationssystemen zwei Charakteristika zu beachten: Zum einen handelt es sich während der Laufzeit derartiger Systeme in der Regel um statische Konfigurationen, d.h., das während des laufenden Betriebes nur Stationen in den Produktivdatenverkehr aufgenommen werden, die bereits in der Konfiguration berücksichtigt sind, und zum anderen existiert eine sehr enge Verknüpfung zwischen der Anwendung und dem Kommunikationssystem. Dies wird z.B. an den Systemen PROFIBUS und FIP deutlich. Beim PROFIBUS sind Kommunikationsobjekte, wie z.B. das *virtuelle Feldgerät* oder *Variable*, zugleich Schnittstelle zur realen Applikation bzw. repräsentieren Anwendungsobjekte. Beim FIP-System sind die *Variablen* als zentrale Kommunikationsobjekte des MPS- und des A/P-Protokolls ebenfalls zugleich Bestandteil der Anwendung.

Das OLCHFA-System hat, bedingt durch seinen Status als Erweiterung eines bereits komplex zu konfigurierenden Basissystems, einen über den des Ursprungssystems hinausgehenden Vorrat an Konfigurationsparametern, denen der Benutzer während der Konfigurierung des Systems korrekte Werte zuweisen muß. Als Beispiele für die zusätzlich zu setzenden Parameter seien die verteilte Uhrensynchronisation, die Definition zeitkritischer Variablen und die Konfigurierung der Ereignisbehandlung genannt [128].

Die Konfiguration eines OLCHFA-Systems wird in einer formalen Sprache beschrieben und in einer Konfigurationsdatei hinterlegt. Die Grammatik der formalen Sprache wird mit *EXF* - Extended FCL (FIP Configuration Language) - bezeichnet und ist eine Erweiterung der zur Beschreibung von Konfiguration von FIP-Netzwerken benutzten Grammatik. Die Konfigurationsbeschreibung für ein OLCHFA-Netzwerk wird mit Hilfe von rechnergestützten Werkzeugen erzeugt und weiterverarbeitet. Dabei entstanden im Rahmen des OLCHFA-Projektes am FAPS die Programme *CONTOUR* und *PreFIPC*. Der Konfigurierungsvorgang ist in Bild 36 dargestellt. Als Notation wurde die Darstellung für *Objektflußdiagramme (Object-flow diagrams)* gemäß *Martin et al.* [129] gewählt. Darin werden beliebige Objekte, z.B. Dateien, als beschriftete Kästen und Aktionen durch Kästen mit abgerundeten Ecken dargestellt. Die Erzeuger-Verbraucher Relation wird durch gerichtete Pfeile versinnbildlicht. Die bereits vorhandenen Werkzeuge zur Konfiguration von FIP-Systemen sind in der Abbildung in dem schraffierten Bereich plaziert. Die am Ende des Konfigurationsprozesses entstandenen Quelltextdateien werden nachfolgend übersetzt und mit den Benutzerapplikationen zu ausführbaren Programmen zusammengebunden.

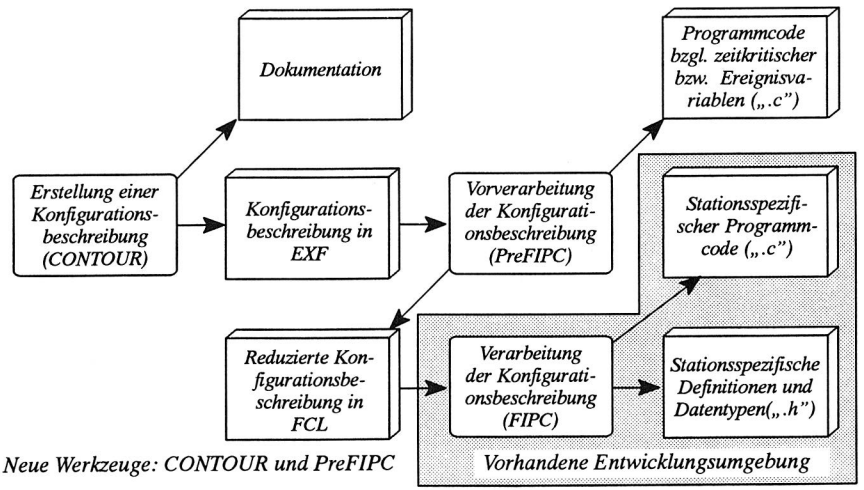


Bild 36: Erzeugung und Verarbeitung einer Konfigurationsbeschreibung

Konfigurationsbeschreibung

Die Grammatik zur Beschreibung von Konfigurationen von OLCHFA-Netzwerken ist eine kontextfreie Grammatik vom Typ LALR(1), d.h., sie basiert auf einem Lookahead (LA) von einem Symbol (1) und ist linksrekursiv (LR). Ein beispielhafter Auszug aus den definierten Ableitungsregeln ist in Bild 37 dargestellt und zeigt den Teil von EXF, der die Attributierung einer produzier-

ten Echtzeitvariablen für die Ereignisbehandlung beschreiben läßt. Die vollständige Beschreibung der EXF-Grammatik ist in [130] enthalten.

```
produced_tc_variable
   : TCSYMBOL tc_produced_variable_attributes
     optional_produced_services tc_variable_values
   ;
tc_produced_variable_attributes
   : optional_identifier optional_pdutype
     variable_period timecritical tc_variable_type
   ;
tc_variable_values
   : default_value monitor_list
   | default_value
   ;
monitor_list
   : monitored monitor_type_list
   ;
monitor_type_list
   : monitor_type_list monitor_type
   | monitor_type
   ;
monitor_type
   : ascending FLOAT HEXNUMBER
   | descending FLOAT HEXNUMBER
   | deadband FLOAT FLOAT HEXNUMBER
   ;
```

Bild 37: Auszug aus den Ableitungsregeln der EXF

Werkzeuge

Zur Erleichterung des Konfigurationsvorganges wurde das Programm CONTOUR (*Configuration Tool for the OLCHFA fieldbus with graphical user interface*) konzipiert und implementiert. CONTOUR erlaubt die schrittweise Erstellung einer Netzwerkbeschreibung, wobei wo immer möglich, wie z.B. bei der Festlegung von Bezeichnern für Variable, automatisch sinnvolle Werte vorgeschlagen werden. Die Darstellung besonderer Eigenschaften des Netzwerkes, z.B. die Art des verwendeten Mediums und die gewählte Datenübertragungsrate, erfolgt an der grafischen Benutzeroberfläche. Ebenso werden besondere Eigenschaften der Station, z.B. ihre Klassifikation als Sensor oder Aktor, der Status des Busarbitrators und die Rolle der Station als Master oder Slave im Verfahren der Uhrensynchronisation („Clock-Master" bzw. „Clock-Slave") veranschaulicht. Bild 38 stellt die Arbeitsoberfläche mit dem Netzwerklayout von CONTOUR dar, wobei beispielhaft ein Netzwerk mit drei Segmenten, die durch Repeater verbunden sind, und insgesamt zehn Stationen konfiguriert ist. Die Eingabemaske zum Setzen der Parameter einer Station ist in Bild 39 dargestellt. Über diese Eingabemaske werden alle spezifischen Parameter einer Station, z.B. ihre Stationsnummer, der in der Kommunikationsanschaltung verwendete ASIC, die Zugehörigkeit zu definierbaren Stationsgruppen, die Rollenzuteilung bezüglich Busarbitrierung und Uhrensynchronisation usw. gesetzt. CONTOUR führt alle aus den Festlegungen eindeutig ableitbaren Aktionen automatisch durch. So werden z.B. für jeden Clock-Master zwei Synchronisationsvariablen definiert und als von der Station produziert gekennzeichnet. Die eindeutige Namensgebung für diese Variablen erfolgt unter Verwendung des Stationsnamens und wird bei dessen Änderung

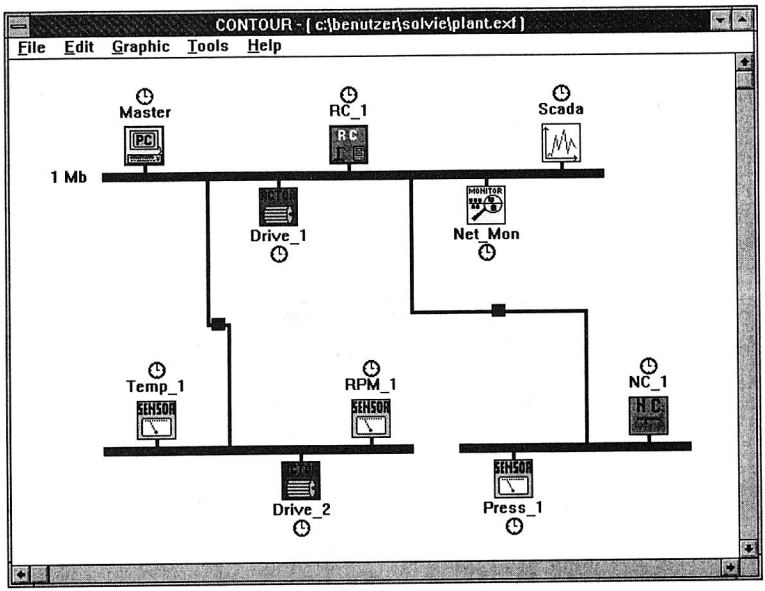
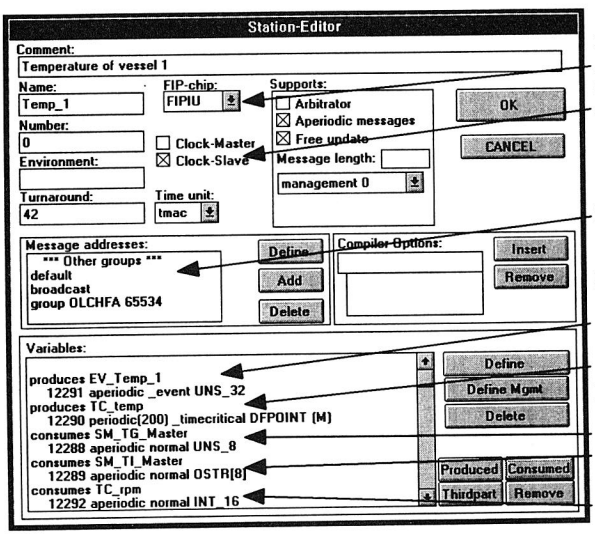


Bild 38: CONTOUR – Benutzeroberfläche: Netzwerk-Layout



Definition der Station *„TEMP_1" mit FIPIU-ASIC* in der Rolle eines *Clock-Slave*

Zugehörigkeit zu Gruppen

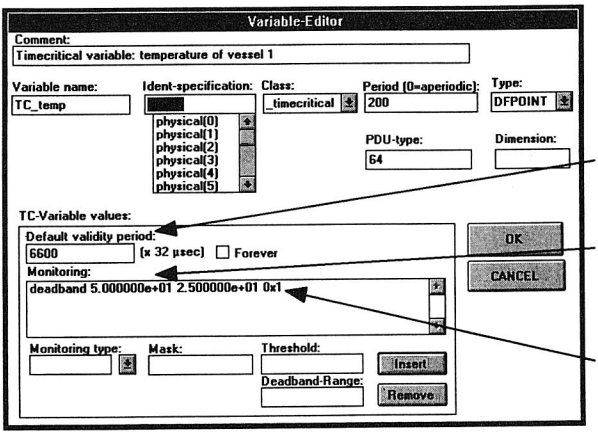
Produziert werden die *Ereignisvariable „EV_TC_TEMP_1"* und die *zeitkritische Variable „TC_TEMP"*

Konsumiert werden die *Synchronisationsvariablen „SM_TG_MASTER", „SM_TI_MASTER"* und die *zeitkritische Variable „TC_rpm"*

Bild 39: Stationseditor von CONTOUR mit Erläuterung wesentlicher Bereiche

konsistent in der gesamten Konfigurationsbeschreibung modifiziert. Als Clock-Slaves konfigurierte Stationen werden - entsprechend dem Vorgehen bei den Mastern - sinngemäß als Konsumenten aller Synchronisationsvariablen deklariert.

Im Stationseditor werden auch die Rollenverteilung einer Station als Produzent oder Konsument von Variablen festgelegt. Die Definition von Variablen findet mittels der diesbezüglichen Eingabemaske statt (vgl. Bild 40), die vom Stationseditor aus aufgerufen wird. Werden obere oder untere Grenzwerte für den Wert einer Variablen festgelegt, so erzeugt CONTOUR automatisch die zur Benachrichtigung über den Eintritt eines Ereignisses potentiell benötigte Ereignisvariable, die dann bereits vollständig vorkonfiguriert ist.



Definition der *Echtzeitvariable „TC_TEMP"*

Die besonderen Attribute betreffen die

Gültigkeitsdauer

und die

festgelegten Grenzwerte

sowie die

Kennung des Ereignisses

Bild 40: Variableneditor von CONTOUR

CONTOUR verfügt zudem über eine Funktion zur Überprüfung der Konsistenz des aktuellen Standes der Konfigurationsbeschreibung, bei der Hinweise auf mögliche Fehler in der Konfiguration gegeben werden, wie z.B., daß eine Variable zwar als konsumiert deklariert ist, aber von keiner Station produziert wird. Enthält eine Konfigurationsbeschreibung zum Zeitpunkt der Beendigung der Bearbeitung derartige Inkonsistenzen, so werden diese von CONTOUR beim Erzeugen der textuellen Beschreibung eliminiert. Bei diesem Arbeitsschritt wird für Zwecke der Dokumentation auch eine textuelle Beschreibung der Konfiguration in Form von Querbezugslisten, Beschreibung von Ereignisbedingungen usw. erzeugt.

Es bestand bei der Entwicklung der Konfigurationswerkzeuge die Zielsetzung, soweit wie möglich auf verfügbare Werkzeuge aufzusetzen. Aus diesem Grunde wurde mit *PreFIPC* ein Präprozessor konzipiert und implementiert, der die zur Abarbeitung der Protokollerweiterungen notwendigen Informationen, wie z.B. die vordefinierten Gültigkeitszeitdauern für zeitkritische Variable, aus einer Beschreibung gemäß der EXF-Grammatik extrahiert und der Anwendung und dem MPS-E Protokoll in einer definierten Form als Ergänzung der Management-Information zur Verfügung stellt. Zudem wird eine Beschreibung des Netzwerkes ohne Berücksichtigung der Erweiterungen gemäß der Grammatik FCL erzeugt, die mit vorhandenen Werkzeugen weiterverarbeitet werden kann. Alle dabei entstehenden Informationen, die zur Laufzeit benötigt werden, werden mit den Applikationen für die einzelnen Stationen zusammengebunden und bilden dann ein aus-

führbares Programm für OLCHFA-Kommunikationsanschaltungen bzw. für PCs. Eine ausführliche Darstellung der Eigenschaften der Werkzeuge findet sich in [131].

4.1.6 Bewertung

Das OLCHFA-System bietet in Bezug auf die dargestellten Anforderungen gegenüber dem im zugrundeliegenden FIP-System einige wesentliche Erweiterungen im Bereich der Integration der Dimension „Zeit" in das Kommunikationssystem und wird bezogen auf die analysierten Anforderungen wie folgt bewertet:

ANFORDERUNG EZ-1: Rechtzeitigkeit

Die Rechtzeitigkeit der Dienstausführung wird wie im unterlagerten FIP-System realisiert und kann damit nur für den periodischen Austausch von Daten gewährleistet werden.

ANFORDERUNG EZ-2: Berücksichtigung von Echtzeit- und Nicht-Echtzeit Kommunikationsanforderungen

Es können Echtzeit- und Nicht-Echtzeit Kommunikationsanforderungen befriedigt werden. Bei den Echtzeit-Kommunikationsanforderungen besteht über die Mechanismen des FIP-Basissystems hinaus die Option, die Dienste des MPS-E Protokolls zu nutzen. Bei Verwendung dieser Dienste wird gewährleistet, daß unter dem Aspekt der limitierten zeitlichen Gültigkeit von Werten von Variablen nur gültige Werte in die verteilte Datenbasis eingebracht und auch nur noch gültige Werte daraus wieder ausgelesen werden können.

ANFORDERUNG EZ-3: Vorhersehbarkeit

Es gelten die für das FIP-System gemachten Aussagen.

ANFORDERUNG EZ-4: Zuverlässigkeit

Das OLCHFA-System an sich weist die gleichen Charakteristika bzgl. der Zuverlässigkeit auf, wie das FIP-System. Eine zusätzliche Fehlerquelle wird durch den zentral gesteuerten Mechanismus der Uhrensynchronisation eingebracht. Die Güte der Synchronisation der lokalen Uhr bezüglich der Uhrzeit des Masters ist jedoch jederzeit verfügbar, so daß bei schlechter oder fehlender Synchronisation dieser Zustand zumindest erkannt werden kann.

ANFORDERUNG EZ-5: Verarbeitung von Echtzeitdaten

Das OLCHFA-System bietet erweiterte Dienste zur Verarbeitung von Echtzeitdaten an. Die entsprechenden Datentypen sind implementiert worden. Alle geforderten Datenkonsistenzattribute werden unterstützt. Die Auswertung der Gültigkeit einer Variablen ist nicht mehr nur an die Mechanismen der Evaluierung des „Promptness"- bzw. „Refreshment"-Status gebunden, sondern ist relativ zur Gültigkeit der Variablen und deren Erzeugungszeitpunkt möglich.

ANFORDERUNG EZ-6: Unterstützung einer verteilten Uhr

Das OLCHFA-System weist eine, mittels Hardware-Unterstützung realisierte, synchronisierte verteilte Uhr auf. Diese erlaubt eine Synchronisation mit einer maximalen Abweichung der Uhrzeiten zweier beliebiger Stationen im eingeschwungenen System im Bereich von 20 µs. Die Synchronisation der lokalen Uhren erfolgt stetig durch Verlangsamen bzw. Beschleunigen des Taktes der lokalen Uhr. Nach der Einschwingphase können die geforderten Bedingungen *PU 1* und *PU 2* eingehalten werden.

ANFORDERUNG VS-1: Benutzergesteuerte Prioritätenvergabe

Zeitkritische Dienste werden grundsätzlich hochprior ausgeführt. Ansonsten gelten die für das FIP-System gemachten Aussagen.

ANFORDERUNG VS-2: Synchronisation von Applikationen

Über den vergleichsweise rudimentären Ansatz zur Synchronisation von Applikationen, der durch das Basissystem FIP angeboten wird, steht mit der netzweit synchronisierten Uhrzeit ein wichtiges Hilfsmittel zur Synchronisation von verteilten Applikationen bereit. Basierend auf der Uhrzeit kann z.B. ein Scheduling von Applikationen autonom in den einzelnen Knoten erfolgen.

ANFORDERUNG VS-3: Scheduling des Kommunikationssystems

Es gelten die für das FIP-System gemachten Aussagen.

ANFORDERUNG VS-4: Breites Spektrum an PDU-Sequenzen

Es gelten die für das FIP-System gemachten Aussagen.

ANFORDERUNG VS-5: Autonomie

Prinzipiell gelten die gleichen Aussagen wie sie für das FIP-System getroffen wurden. Die spezielle Implementierung des OLCHFA-Systems als leistungsfähige Stationen mit Ressourcen für die lokale Datenverarbeitung stellt eine dafür ausreichende Basis dar. Die Verfügbarkeit der netzweit synchronisierten Zeit und der Einsatz eines Echtzeit-Betriebssystems erlauben zudem ein Scheduling von lokalen Applikationen in globaler Synchronisation ohne externen Anstoß, wobei allerdings keine Unterstützung des Schedulings durch das Kommunikationssystem geboten wird. Der Mechanismus der Ereignisbehandlung erlaubt die lokale Erkennung von aufgetretenen Ereignissen und einen autonom durchgeführten, verteilten Aufruf der Behandlungsfunktionen.

ANFORDERUNG VS-6: Komfortable Kommunikationsdienste

Über die Dienste des MPS-Protokolls des FIP Systems hinaus werden im OLCHFA-System Dienste zur Behandlung zeitkritischer Daten angeboten. Zusätzlich sind das Ereignis-Management und der Dienst `A-INVALID` hinzugekommen. MMS-ähnliche Dienste aus dem SubMMS-Protokoll werden nicht zur Verfügung gestellt.

ANFORDERUNG VS-7: Berücksichtigung von ereignis- und zustandsgesteuerter Kommunikation

Beide Kommunikationstypen werden wie im FIP-System unterstützt. Die Erweiterungen erlauben dabei eine Behandlung von Echtzeitvariablen sowohl in der zustands- als auch in der ereignisgesteuerten Kommunikation. Zur Unterstützung der ereignisgesteuerten Kommunikation ist die Ereignisbehandlung vorgesehen. Sie beinhaltet eine für den Benutzer transparente Prüfung auf das Vorliegen eines Ereignisses und eine ebenso transparente Ereignisbehandlung.

ANFORDERUNG VS-8: Unterstützung von Replikationsmechanismen für Daten

Es gelten die für das FIP-System gemachten Aussagen.

ANFORDERUNG MM-1 - MM-3: Multimedia-Integration

Es gelten die für das FIP-System gemachten Aussagen.

4.1.7 Übertragbarkeit der Erweiterungen

Ein Teil der Erweiterungen, die im OLCHFA-System für das Feldkommunikationssystem FIP konzipiert und realisiert worden sind, läßt sich auf andere Feldkommunikationssysteme, z.B. PROFIBUS, übertragen. Dabei existieren bei jedem System durch dessen Konzeption bedingte Probleme bei der Erweiterung. Dies sei am Beispiel des PROFIBUS kurz skizziert:

PROFIBUS erlaubt prinzipiell die Definition beliebiger Variablentypen. Die Dienste zum Zugriff auf Variablenobjekte sind jedoch nicht geeignet, die zeitliche Gültigkeit von Variablen auszuwerten (vgl. auch Kap. 3.4.2). Die Integration zusätzlicher Dienste in PROFIBUS gestaltet sich bereits in der konzeptionellen Phase als relativ komplex. So sieht PROFIBUS z.B. beim Aufbau von Verbindungen zwischen Stationen das Aushandeln des Verbindungskontextes vor, der u.a. auf dem Austausch von Informationen über das unterstützte Dienstespektrum beruht. Für diesen Dienst ist eine Protokolldateneinheit definiert, die keine benutzerspezifischen Erweiterungen zuläßt. Ein Lösungsweg besteht darin, PROFIBUS zusätzlich um neue administrative Dienste zum Verbindungsaufbau zu erweitern. Die Sicherung der Interoperabilität mit nicht erweiterten PROFIBUS-FMS Systemen kann mittels in einer erweiterten Kommunikationsbeziehungsliste gehaltener Attribute vorgenommen werden, in der neben den aus PROFIBUS-FMS bekannten Informationen auch solche über den Typ des Kommunikationspartners enthalten sind. Die Einführung neuer Objekte, z.B. das der Echtzeit-Variable, bedeutet keine besonderen Anforderungen, sofern eine Beschränkung auf statisch definierte Objekte erfolgt. Die Definition der Echtzeit-Variable basiert auf der regulären Beschreibung einer Variablen und beinhaltet zusätzlich die geforderten Konsistenzattribute.

Auf Basis derartiger Erweiterungen von PROFIBUS bezüglich administrativer Dienste können zusätzliche Produktivdienste eingeführt werden, die auf den Echtzeitvariablen wirken und eine Überprüfung der zeitlichen Gültigkeit sowie der Konsistenzattribute vornehmen.

4.2 Multimedia-Datentransfer über PROFIBUS

Im Rahmen laufender Arbeiten wurden am Lehrstuhl FAPS Konzepte zur Übertragung von Multimedia-Daten über das Feldkommunikationssystem PROFIBUS entwickelt und die zugehörigen Implementierungen vorgenommen. Dabei sind mit der Übertragung von Audio-, Video- und Standbildinformationen alle für den prozeßnahen Bereich wesentlichen zusätzlichen Informationstypen behandelt worden. In allen Implementierungen, die nachfolgend kurz skizziert werden, wurde dabei auf die Verwendung des FDL-Protokolls als Schnittstelle zur Sicherungsschicht von PROFIBUS zurückgegriffen, da sich bei Messungen des erzielbaren Datendurchsatzes mit den verfügbaren Implementierungen von FMS Durchsatzprobleme deutlich abzeichneten.

4.2.1 Audio-Datentransfer

Auf Basis von PROFIBUS-FDL wurde eine Applikation zur Übertragung von Audiodaten im Bereich zwischen 0 und 3000 Hz konzipiert und realisiert. Dieses System besteht in Hardware aus einem PC mit handelsüblicher Anschaltung an das Feldkommunikationssystem und Klangwiedergabebaugruppe sowie der entwickelten Audio-Feldbaugruppe, die an eine intelligente PROFIBUS-Baugruppe (Typ IM 318 ME) angeschlossen werden kann. Die Audio-Baugruppe beinhaltet einen Mikroprozessor (PIC 16C57) und erlaubt die Aufnahme und Wiedergabe von Audiodaten, die mit einer Rate von 8 kHz abgetastet und mit 8 Bit quantisiert werden. Durch die in Software realisierte Kodierung nach dem ADPCM-Verfahren werden die aufgenommenen Daten im Verhältnis 2:1 komprimiert. Dieses Kodierungsformat wird durch die Klangwiedergabebaugruppe im PC ebenfalls unterstützt. Die resultierende Busbelastung beträgt ca. 32 kbps.

Die zugehörige Software wurde auf zwei Anwendungsfälle ausgerichtet: Die Steuerung der Aufnahme von Audio-Daten zum Zwecke der Überwachung technischer Prozesse und für die Bediener-Bediener-Kommunikation. Für letzteren Anwendungszweck wurde eine *Voice-Mailbox*-Funktionalität realisiert, die eintreffende, nicht durch den Bediener bestätigte Meldungen spei-

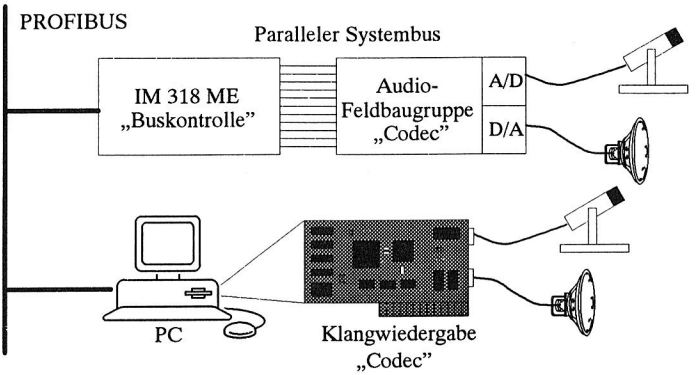


Bild 41: Struktur der Applikation zu Audiodatenübertragung

chert und die aktuelle Anzahl gespeicherter Meldungen am PC textuell unter Angabe der Uhrzeit des Eintreffens der letzen Nachricht und an der Feldbaugruppe durch einen entsprechenden Blinkrhythmus einer Leuchtdiode anzeigt. Die Nachrichten können dann später abgerufen werden.

In beiden Fällen fungiert die Applikation, die auf dem PC abgearbeitet wird, als Client-Applikation, die das gesamte System steuert. Pro zyklisch gesendetem Datenpaket können – beschränkt durch die zur Verfügung stehende Implementierung von PROFIBUS – 32 Byte Nutzdaten übertragen werden. Davon werden zwei Byte zur Übertragung von Steuerinformationen, die restlichen Bytes zur Übertragung von kodierter Audio-Information genutzt, wobei der SRD-Dienst des Protokolls der Sicherungsschicht von PROFIBUS zur Anwendung kommt [59].

4.2.2 Bewegtbild-Datentransfer

Auf der Basis des gleichen Feldbussystems wurde eine Applikation zur Übertragung von gemäß dem Standard H.261 kodierten Bewegtbild-Informationen konzipiert und realisiert. Die Kodierung der anfallenden Bewegtbildinformationen wird mittels einer diesbezüglichen, handelsüblichen PC-Erweiterungsbaugruppe vorgenommen. Diese Baugruppe unterstützt die Kodierung von Bewegtbildern nur für eine anvisierte Datenrate von 64 kbps, dafür aber sowohl im Format CIF als auch im Format QCIF. Die Dekodierung der Daten wird per Software vorgenommen.

Der Schwerpunkt der Arbeit lag auf der Konzeption und Implementierung des Servers, der durch das System gebildet wird, das als Quelle der kodierten Bewegtbild-Daten dient. Dieser stellt Funktionen zur Abfrage des momentanen Status sowie zum Starten und Stoppen der Kodierung und Übertragung von Bewegtbild-Informationen zur Verfügung. Logisch gesehen existiert somit ein Kontroll- und ein Datenkanal zwischen dem Server und dem bzw. den Client(s). Die Struktur der Applikation ist in Bild 42 dargestellt [7].

Der durch den Koder erzeugte Datenstrom wird im Server für die Übertragung über PROFIBUS paketiert, wobei aus Gründen der Erzielung eines möglichst hohen Durchsatzes Pakete maximaler Länge (244 Byte) gesendet werden, auch wenn damit durch die notwendige Pufferung eine größere Verzögerung in Kauf genommen werden muß. Die wichtigsten Bedienungselemente einer protoypischen Client–Applikation zur Steuerung der Kamera und Anzeige des Bildes sind in Bild 43 dargestellt.

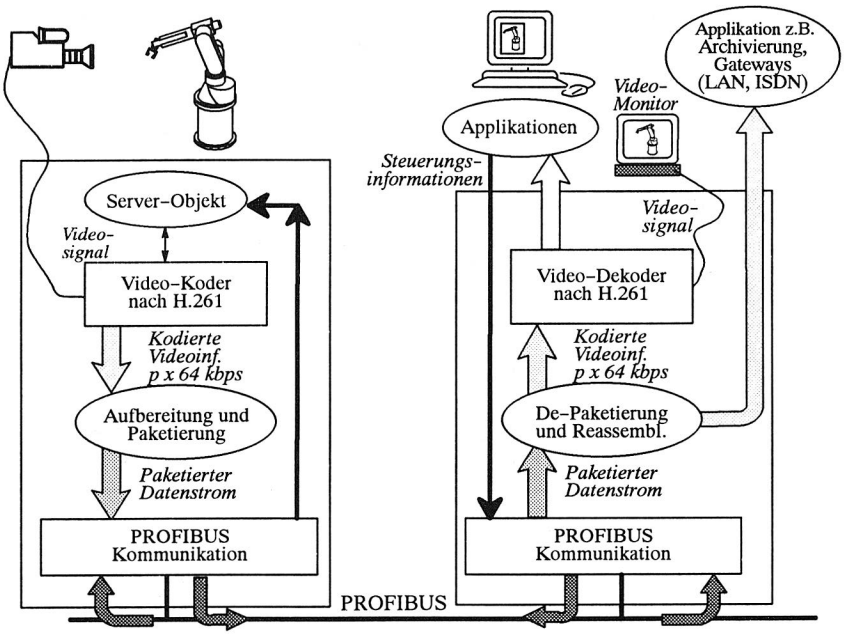


Bild 42: Anwendungs- und Kommunikationsarchitektur der Bewegtbildübertragung über PROFIBUS

Das Ziel, die Videodaten von der Quelle aus im Multicast an mehrere Empfänger gleichzeitig senden zu können, bedeutet bei Verwendung des PROFIBUS, daß die Datenquelle bezüglich des Buszugriffes als Master fungieren muß. Um auch den bezüglich des Sendens auf dem Bus passiven Slave-Stationen die Möglichkeit der Übernahme der Rolle des Clients zu ermöglichen, fragt der Server alle angeschlossenen Stationen zyklisch bezüglich des Vorliegens von Übertragungswünschen ab. Zur Übertragung der einzelnen, zu einem Bewegtbild-Datenstrom gehörigen Datenpakete, wird der SDN-Dienst von PROFIBUS genutzt. Es empfiehlt sich, bei der Verwendung des Systems auf eine Mono-Master-Konfiguration zurückzugreifen, da anderenfalls Teile des Datenstromes verzögert werden, wenn die Sendeberechtigung einer anderen Masterstation zugeteilt ist.

4.2.3 Aufnahme und Übertragung von Standbildern

Die Übertragung von Standbildern wurde in PROFIBUS ebenfalls unter direkter Nutzung der Dienste des FDL-Protokolls konzipiert und realisiert. Die bereitgestellte Funktionalität erlaubt es, an einer Datenquelle für Standbilder, z.B. einer Videokamera mit nachgeschaltetem Bilderfassungssystem (*Frame Grabber*) die Erzeugung eines Standbildes und die Kodierung mittels des JPEG-Algorithmus in einer durch den Benutzer wählbaren Qualitätsstufe vorzunehmen. Die Übertragung der erzeugten Daten erfolgt blockweise von der Datenquelle zu der anderen Station. Zur Sicherung der Datenübertragung wird ein einfaches Protokoll mit Verwendung von Sequenznummer und erneuter Übertragung eines Datenpaketes im Fehlerfall verwendet [7].

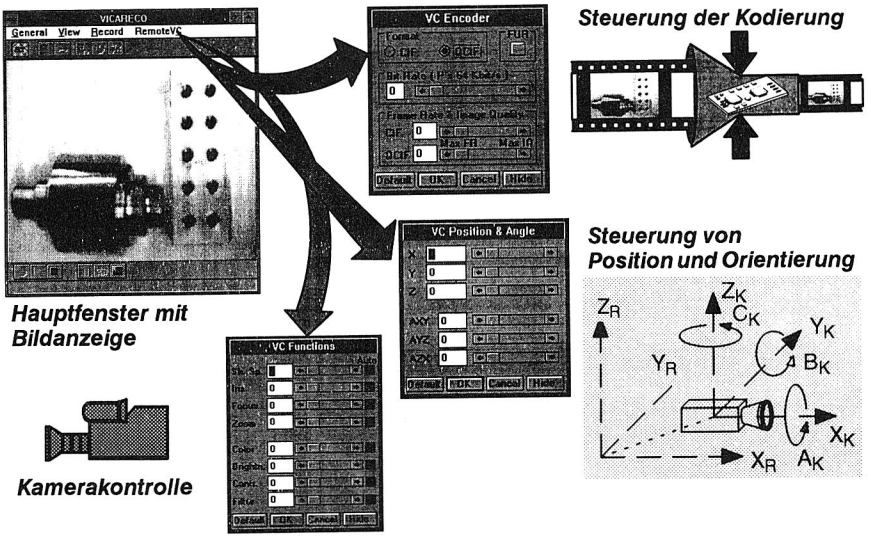


Bild 43: Prototypischer Client zur Ansteuerung einer Kamera über PROFIBUS

4.2.4 Bewertung

Die vorgestellten Ansätze, die Übertragung von Multimedia-Daten über PROFIBUS vorzunehmen, berühren die Anforderung bezüglich der Unterstützung der Steuerung von Aufnahmesystemen durch das Kommunikationssystem. Bezüglich der weiteren Anforderungen gelten die bereits in Kapitel 3.4.2 für PROFIBUS getroffenen Aussagen.

ANFORDERUNG MM-2: Unterstützung von Aufnahmesystemen durch das Kommunikationsprotokoll

In den drei Implementierungen wurden Funktionen zur Unterstützung der Steuerung von Aufnahmesystemen über das Kommunikationssystem in Form der Bereitstellung entsprechender Dienste realisiert, so daß diese Anforderung als erfüllt gilt.

4.3 Zusammenfassung

An zwei ausgewählten, verschiedenen Systemen wurde die Berücksichtigung der aufgestellten Anforderungen durch Nutzung von dessen Basisfunktionalität konzipiert und realisiert. Dabei wurde das OLCHFA-System konsequent auf die Erfüllung neuer Anforderungen im Bereich der Unterstützung der Dimension „Zeit" durch ein Feldkommunikationssystem ausgerichtet und zeigt entsprechende Verbesserungen gegenüber dem Basissystem FIP auf. Die Architektur des Gesamtsystems wird durch Konfigurierungswerkzeuge ergänzt, die es erlauben, auch die Besonderheiten des Systems, z.B. die Ereignisbehandlung und die limitierte zeitliche Gültigkeit von Echtzeitvariablen zu parameterisieren. Wesentliche Bestandteile des Konzeptes zur Behandlung von zeitkritischen Prozeßvariablen können auf andere Feldbussysteme übertragen werden, wie am Beispiel PROFIBUS skizziert wurde. Die Tauglichkeit des OLCHFA-Systems für industrielle Anwendungen wurde in zwei Pilotapplikationen nachgewiesen [132]. Eine vergleichende Darstellung der

Anforderung \ System	PROFIBUS FMS	PROFIBUS DP	FIP	Erw. Systeme OLCHFA	Erw. Systeme PB+MM
EZ-1: Rechtzeitigkeit	◔	◐	◐	◐	◐
EZ-2: Echtzeit und nicht-Echtzeit-Kommunikation	◐	◐	◕	◕	◐
EZ-3: Vorhersehbarkeit	◕	●	◐	◐	◕
EZ-4: Zuverlässigkeit	◐	◐	◐	◐	◐
EZ-5: Verarbeitung von Echtzeitdaten	○	○	○	●	○
EZ-6: Unterstützung einer verteilten Uhr	○	○	○	●	○
VS-1: Benutzergesteuerte Prioritätenvergabe	◔	○	◔	◔	◔
VS-2: Synchronisation von Applikationen	○	◐	◐	◕	○
VS-3: Scheduling des Kommunikationssystems	◐	◐	◐	◐	◐
VS-4: Breites Spektrum an PDU-Sequenzen	◐	◐	●	●	◐
VS-5: Autonomie	◐	○	◔	◐	○
VS-6: Komfortable Kommunikationsdienste	●	○	◕	◐	○
VS-7: Ereignis- und zustandsgesteuerte Kommunikation	●	●	●	●	●
VS-8: Replikationsmechanismen für Daten	◐	○	●	●	○
MM-1: Übertragung kontinuierlicher und diskreter Medien	◔	◐	◐	◐	◐
MM-2: Unterstützung von Aufnahmesystemen	◔	○	○	○	●
MM-3: Synchronisation von Datenströmen	◔	◕	◕	◕	◕

● Vollständig erfüllt ◕ – ◔ Teilweise erfüllt ○ Nicht erfüllt

Bild 44: Übersicht über die Bewertung der Systeme PROFIBUS, FIP, OLCHFA und des erweiterten PROFIBUS (PB+Multimedia (MM))

Beurteilungen der Feldkommunikationssysteme PROFIBUS und FIP, des OLCHFA-Systems und des Ansatzes, Multimedia-Daten über PROFIBUS zu übertragen, ist in Bild 44 zu finden.

Zusammenfassend kann festgestellt werden, daß die Erweiterung vorhandener Systeme in einigen Teilbereichen durchaus ein erfolgversprechendes Vorgehen sein kann, eine Berücksichtigung aller Anforderungen jedoch nicht erreichbar ist. Die durch die Konzepte der vorhandenen Systeme gegebenen Einschränkungen, z.B. die Inflexibilität des FIP-Systems bei Änderungen der Konfiguration und die nur unzureichende Unterstützung von Basismechanismen für die Echtzeitkommunikation durch PROFIBUS, erweisen sich diesbezüglich als zu schwerwiegend.

5 OSIRIS: Übertragungsorientierte Kommunikationsprotokolle

Die angestellten Untersuchungen haben gezeigt, daß existierende Feldbussysteme die im dritten Kapitel dargestellten Anforderungen nur teilweise erfüllen können. Auch die Erweiterung von vorhandenen Systemen erweist sich, wie in Kapitel 4 gezeigt wurde, als nicht unproblematisch und als trotzdem nur teilweise erfolgversprechend.

In diesem Kapitel wird deshalb das Konzept eines neuen Feldkommunikationssystems vorgeschlagen, das auch auf die Erfüllung der in Abschnitten 3.1 bis 3.3 dargestellten Anforderungen hin ausgerichtet ist. Dieses System wird mit *OSIRIS* (Open, Service-Integrating Real-Time fieldbus System) bezeichnet, da es sowohl die Anforderungen bezüglich Echtzeitkommunikation als auch bezüglich der Dienstintegration für die Unterstützung der Aufnahme und des Austausches multimedialer Daten berücksichtigt. In dieser Arbeit können dabei nicht alle Bestandteile eines derartigen, durchaus komplexen, Systementwurfs gleichermaßen vertieft werden. Vielmehr werden die wesentliche Aspekte hervorgehoben, die bezüglich der Erfüllung der Anforderungen Schlüsselpositionen einnehmen.

5.1 Gesamtkonzept und Zielsetzung

Um den aufgestellten Anforderungen gerecht zu werden muß OSIRIS über entsprechend ausgelegte Kommunikationsprotokolle mitsamt der zugehörigen Dienste verfügen, sowie eine systemweit ausreichend genau synchronisierte Uhrzeit bereitstellen.

In Feldkommunikationssystemen spielen die Protokolle der Sicherungsschicht eine wesentliche Rolle bei der Erbringung der gesamten Funktionalität. In OSIRIS wird in der Sicherungsschicht der Grundstein für die Übertragung von Daten in Echtzeit gelegt, die sowohl für den Prozeßdatenaustausch als auch für die isochrone Übertragung von multimedialen Datenströmen benötigt wird. Zudem wird vom Protokoll der Sicherungsschicht von OSIRIS die Bereitstellung von Basisfunktionalität zur Realisierung einer synchronisierten Uhrzeit erbracht. Als Medienzugriffsprotokoll (MAC, Medium Access Control) kommt in OSIRIS ein dem DQDB (Distributed Queue Dual Bus) [133] teilweise nachempfundenes Protokoll zum Einsatz: *DQDFB* (Distributed Queue Dual Field-Bus). Von DQDB werden dabei im wesentlichen die Topologie, die Architektur eine Station auf Ebene der MAC-Teilschicht der Sicherungsschicht, die Berücksichtigung von periodischem und aperiodischem Datenaustausch mittels der in einem festen Takt gesendeten Übertragungsrahmen und das Verfahren der verteilten Warteschlange übernommen. Die Festlegungen der Dienste, Protokolldateneinheiten und des Verfahrens zum Austausch von periodischen Daten sind entsprechend der Belange des Prozeß- und Multimedia-Datenaustausches mittels Feldkommunikationssystemen gewählt. Der durch den Austausch von Übertragungsrahmen gegebene feste Takt wird als Basis für die Uhrzeitsynchronisation genutzt. DQDFB wird zusammen mit der Architektur einer DQDFB-Station und den definierten Diensten in Kapitel 5.3 vertiefend betrachtet.

Die Bereitstellung eines Replikationsmechanismus durch Feldkommunikationssysteme ist für Steuer- und Regelungsapplikationen von großer Bedeutung. In OSIRIS wird dieser Mechanismus durch eine verteilte Datenbasis, wie sie aus FIP und ansatzweise auch aus PROFIBUS bekannt ist, unter Verwaltung des neu entwickelten *Distributed Database, Realtime Services, Stream Support Link*- Protokolls (DRSLP) bereitgestellt. Dieses Protokoll der LLC (Logical Link Control)-Teilschicht der Sicherungsschicht wird zusammen mit den zugehörigen Diensten und Mechanismen in Kapitel 5.4 näher vorgestellt. Es beinhaltet Funktionalität zum Austausch von verschiedenen

Klassen von Daten. Im einzelnen werden dabei unterschieden: Prozeßvariable, Dateneinheiten aus multimedialen Datenströmen und Nachrichten.

Die geforderte Erbringung einer komfortablen Kommunikationsfunktionaltität auf Basis der von der Sicherungsschicht angeboten Dienste setzt Mechanismen zur Verwaltung logischer Verbindungen voraus. Diese Aufgabe wird in der für OSIRIS festgelegten Protokollarchitektur (vgl. Bild 45) durch das sog. *Lower Layer Interface (LLI)* (vgl. Kapitel 6.1) erbracht. Daneben werden durch das LLI die Dienstanforderungen der aufsetzenden Protokolle auf die Funktionalität von DRSLP abgebildet. Das LLI wird als Bestandteil der Anwendungsschicht geführt und schafft für seine Dienstbenutzer eine weitgehende Transparenz bezüglich der unterschiedlichen Klassen von Datenaustauschmechanismen. Bei der Definition des LLI wurden die entsprechenden Konzepte von PROFIBUS berücksichtigt und in einer angepaßten und vereinfachten Form eingesetzt.

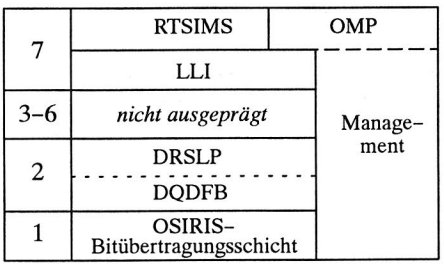


Bild 45: OSIRIS Protokollarchitektur im ISO-OSI Basisreferenzmodell

Für die Schichten 3-6 des ISO-OSI Basisreferenzmodells sind – wie in Feldkommunikationssystemen üblich – keine Ausprägungen von Protokollen der einzelnen Schichten vorgesehen.

Aus Benutzersicht kommt dem Protokoll der Anwendungsschicht eine besonders große Bedeutung zu. In diesem Teil des Kommunikationssystems wird die für den Benutzer direkt nutzbare Funktionalität erbracht. Für OSIRIS besteht hier insbesondere die Forderung nach einem komfortablen und ausdrucksfähigen Protokoll zur Steuerung von Feldgeräten und von Aufnahmesystemen für multimediale Informationen und zur Übertragung der dabei anfallenden Daten. Zu diesem Zweck steht in OSIRIS das auf dem LLI aufsetzende Protokoll RTSIMS (*Real-Time Service Integrating Message Specification*) zur Verfügung, das in Kapitel 6.2 näher beschrieben wird. Abgesehen von der Integration von Multimedia-Kommunikationsunterstützung beinhaltet RTSIMS über die für FMS (PROFIBUS) vorgesehenen Konzepte hinaus Mechanismen, die eine weitergehende Autonomie, eine ausdrucksfähigere Synchronisation der Applikation und die Behandlung von Echtzeitdaten erlauben. Dabei sind in OSIRIS alle Stationen gleichberechtigt. Ein Master-Slave Verhältnis, wie z.B. in PROFIBUS, besteht nicht. Durch diese Gleichberechtigung wird insbesondere die Unterstützung einer lokalen Autonomie ohne durch das Kommunikationssystem bedingte funktionale Einschränkungen ermöglicht. Sehr einfache Stationen, z.B. Sensoren mit rein binärer Information, sollen über entsprechende Subsysteme, wie z.B. ASI, an OSIRIS angeschlossen werden.

Neben der Datenkommunikation zwischen Applikation verlangt auch die Verwaltung des Kommunikationssystems an sich nach entsprechender Berücksichtigung. Das zu diesem Zweck in OSIRIS definierte Systemmanagement wird in Kapitel 7.1 vorgestellt. Es umfaßt u.a. Mechanis-

men zur Verwaltung der Konfiguration und die zur Synchronisation der Uhren notwendigen Objekte und Funktionen.

Auf eine Festlegung der genauen Spezifika der Bitübertragungsschicht wird in dieser Arbeit verzichtet. Die grundlegenden funktionalen Anforderungen werden jedoch im nachfolgenden Abschnitt dargestellt.

5.2 Funktionalität und Protokoll der Bitübertragungsschicht

OSIRIS basiert auf der Verwendung zweier gegenläufiger, getakteter, gleichartiger Busse. Alle Stationen sind aktiv an diese Busse angeschaltet. Damit sind die Vorteile einer größeren Ausdehnung des Gesamtsystems und der Datenübertragung im Vollduplex-Betrieb verbunden; es muß jedoch eine prinzipbedingte Latenzzeit in Kauf genommen und in die Systemkonzeption mit einbezogen werden. Als Medien können drahtgebundene Systeme oder Lichtwellenleiter zum Einsatz kommen.

Um auch bei Ausfall einer Station die Kommunikation zwischen den verbliebenen Stationen weiterhin zu ermöglichen, wird die Ankopplung der Stationen an das Medium über Multiplexer vorgenommen, mit deren Hilfe die Zugriffseinheit der Station im Falle eines Fehlers überbrückt werden kann. Zur Steuerung des Multiplexers wird ein Statussignal der Zugriffseinheit, in der die Buszugriffsfunktionen abgewickelt werden, verwendet.

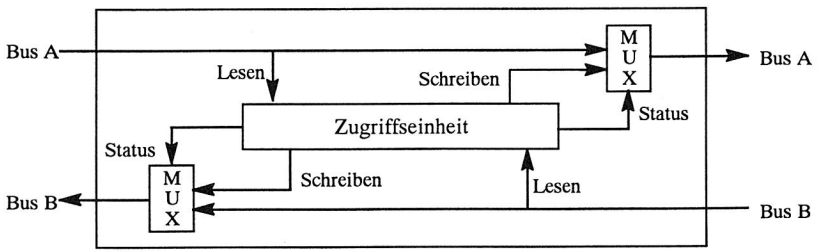


Bild 46: Busankopplung einer Station

Das DQDFB-Protokoll und das Systemmanagement benötigen von der Bitübertragungsschicht die Bereitstellung von Übertragungsdiensten. Diese Dienste stellen samt ihrer Parameter eine funktionale Untermenge der Dienste dar, die in der Norm IEC 1158-2 [134] für Feldkommunikationssysteme spezifiziert sind. Die Reduzierung des Dienst- und Parameterumfangs wird durch die speziellen Charakteristika von OSIRIS, insbesondere die Verwendung einer festen Rahmengröße möglich. Zur Übertragung eines Oktetts wird der Dienst `Ph-DATA` benötigt. Die Parameter der Dienstelemente sind in Tabelle 6 dargestellt.

Parametername	req	ind
Argument	**M**	**M**
Octet	M	M
Class	M	M

Tabelle 6: Parameter der Dienstelemente des Dienstes `PH-DATA`

Das Dienstelement `Ph-DATA.req` erlaubt die Übergabe eines beliebigen Oktetts an die Protokollinstanz der Bitübertragungsschicht im Parameter `octet`. Für den Parameter `class` stehen

die Alternativen „Start-of-activity" und „Data" zur Verfügung. Mittels „Start-of-activity" wird angezeigt, daß das übergebene Oktett den Beginn eines Übertragungsrahmens darstellt, das von genau 35 Oktetten des Typs „Data" gefolgt wird, die mittels der konsekutiven `Ph-DATA.req` Dienstanforderungen übergeben werden. Eine eingetroffene Übertragungseinheit wird mit Hilfe des Dienstes `Ph-DATA.ind` an die DQDFB-Protokollinstanz übergeben. Der Wert des Parameters „Class" kann auf unterschiedliche Weise evaluiert werden. Dies kann durch Überwachung der Ruhezeit auf dem Medium oder - je nach Ausprägung der Bitübertragungsschicht - z.B. auf Basis einer zu Beginn der Übertragung eines Übertragungsrahmens gesendeten speziellen Kennung, beispielsweise einer Kodeverletzung, geschehen.

Die Bitübertragungsschicht stellt der DQDFB-Schicht ihre Dienste an zwei Dienstzugangspunkten (SAP, Service Access Points) zur Verfügung. Jeder dieser Dienstzugangspunkte stellt die Schnittstelle zu einer Vollduplex-Verbindung zu einer benachbarten Station dar (vgl. Bild 47).

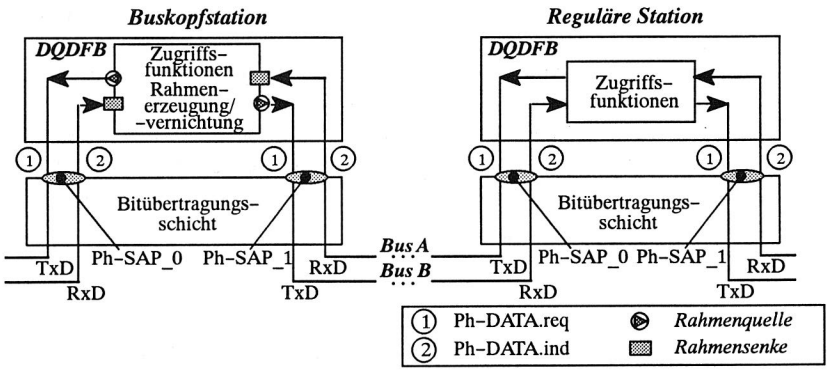


Bild 47: Zusammenhang zwischen Sende- Empfangsdiensten

Erste prototypische Implementierungen von DQDFB basieren in der Bitübertragungsschicht auf der RS-232-C Spezifikation bzw. auf von den verwendeten Transputern bereitgestellten sog. *Links* mit nachgeschalteter elektrischer Signalverstärkung.

5.3 Funktionalität und Protokoll zur Zugriffssteuerung

Vom Medienzugriffsprotokoll der Sicherungsschicht wird gefordert, daß dieses Mechanismen für den periodischen und den aperiodischen Austausch von Dateneinheiten zur Verfügung stellt. Bezüglich des periodischen Austausches von Daten beinhaltet dies auch die Forderung nach Isochronität. Die Länge der dabei ausgetauschten Informationseinheiten soll den Anforderungen aus dem prozeßnahen Bereich genügen, d.h., den Bereich bis zu wenigen hundert Byte abdecken.

DQDFB erfüllt diese Anforderungen wie folgt: Der periodische Austausch von Dateneinheiten basiert auf der Verwendung von reservierten Übertragungsrahmen, den sog. PA (Pre-Arbitrated)-Rahmen. Deren rechtzeitige Erzeugung und Versendung obliegt der Buskopfstation (BKS, vgl. Kapitel 5.3.5), die in DQDFB diesbezüglich eine zentrale Rolle einnimmt. Übertragungsbandbreite, die nicht für den periodischen Austausch von Informationseinheiten genutzt wird, steht für die aperiodische Datenübertragung zur Verfügung. Die Daten werden dabei hoch- oder niederprior in sog. QA (Queued Arbitrated)-Rahmen übertragen. Das Zugriffsverfahren auf die

verfügbaren QA-Rahmen entspricht weitgehend dem von DQDB und wird in Kapitel 5.3.3 kurz erläutert.

DQDFB stellt nur einen unbestätigten Datentransfer zur Verfügung. Werden Bestätigungen über den Erfolg einer Datenübertragung benötigt, so ist dies durch Protokolle der höheren Protokollschichten zu realisieren.

5.3.1 Grundprinzipien des Zugriffsverfahrens

Nachfolgend werden die wesentlichen, zum Verständnis der nachfolgenden Ausführungen notwendigen, Grundprinzipien von DQDFB dargestellt.

Getakteter Bus

OSIRIS basiert auf der Verwendung zweier gegenläufiger, getakteter Busse, auf denen Rahmen einer Länge von 36 Byte in einem festen zeitlichen Abstand gesendet werden. Die Länge von 36 Byte stellt dabei einen günstigen Kompromiß zwischen der für die Uhrzeitsynchronisation notwendigen hohen Rahmenrate, dem Verhältnis von der Menge von Nutzdaten zu der Menge von Protokollinformation pro Übertragungsrahmen und der im Bereich der Feldkommunikation benötigten - typischerweise geringen - Nutzdatenlänge dar.

Bedingt durch das Zugriffsverfahrens von DQDFB muß zwischen zwei aufeinanderfolgenden Übertragungsrahmen eine Ruhezeit von mindestens 24 Bitzeiten eingehalten werden, die als *Latenzzeit* t_{Lat} bezeichnet wird (vgl. Bild 48). Die Rahmenrate f_R ist außer von der Latenzzeit auch

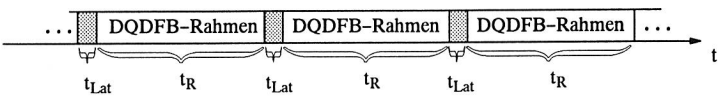


Bild 48: Zeitliche Aufeinanderfolge der DQDFB-Rahmen auf dem Medium

von der Anzahl ggf. benötigter Start- und Stoppbits pro Oktett ($L_{Ü}$) und der Datenübertragungsrate $f_{Ü}$ abhängig. Bei gegebener Datenübertragungsrate, Latenzzeit und $L_{Ü}$ lassen sich die Bitzeit t_{Bit}, die Rahmenzeit t_R, und die Rahmenrate wie folgt berechnen:

$$t_{Bit}(f_{Ü}) = \frac{1}{f_{Ü}} \tag{4}$$

$$t_R(L_{Ü},\ t_{Bit}) = 36 \times (8 + L_{Ü}) \times t_{Bit} \tag{5}$$

$$f_R(t_R,\ t_{Lat}) = \frac{1}{t_R + t_{Lat}} \tag{6}$$

Unter den Annahme, daß $L_{Ü}$=0, wie es bei der Verwendung eines synchronen Verfahrens zur Übertragung von Oktetten, wie z.B. der Manchester-II Kodierung, der Fall ist, bzw. $L_{Ü}$=2, z.B. bei Verwendung eines asynchronen Übertragungsverfahrens mit einem Start- und einem Stoppbit, und t_{Lat} = 24 bzw. t_{Lat} = 32 Bitzeiten, ergeben sich durch Variation der Bitrate die in Bild 49 dargestellten Rahmenraten. Die Werte für t_{Lat} sind dabei als minimaler Wert bzw. unter Berücksichtigung eines Zuschlages von acht Bitzeiten zur Modellierung des Zeitbedarfs für interne Verarbeitungsschritte gewählt.

Für die korrekte Funktion des Systems ist es unabdingbar, daß alle Stationen die Rahmen mit der gleichen Rate verarbeiten. Die Latenzzeit einer jeden Stationen wird deshalb durch das Maximum

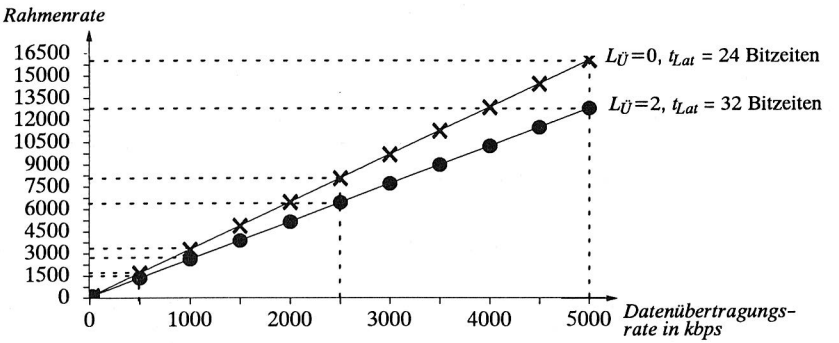


Bild 49: Rahmenrate in Abhängigkeit von $f_Ü$ und $L_Ü$ und t_{Lat}

der Latenzzeiten der einzelnen Stationen festgelegt. Die Erzeugung und Vernichtung der Rahmen wird durch die Buskopfstation vorgenommen. Jeder Rahmen wird von der Erzeugung bis zur Vernichtung vollständig durch das gesamte System geleitet, wobei sich Typ und Inhalt ändern können.

Topologie und Stationsadressierung

DQDFB verwendet - bezogen auf die für DQDB vorgesehenen Topologiealternativen - die geschlossene Topologie. Damit existiert die Buskopfstation als zentrale Station, die, wie in Bild 50 dargestellt, für beide Busse sowohl als Rahmenquelle als auch als Rahmensenke dient.

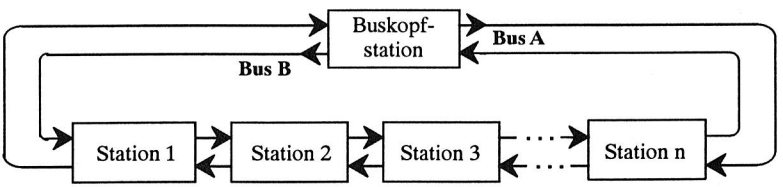


Bild 50: Topologie des OSIRIS-Systems

Stationen haben eine feste Adresse, die während der Konfigurierung festgelegt wird. In einem Netzwerk sind die Stationsadressen 0 bis 127 zugelassen. Zudem ist die Bildung logischer Gruppen vorgesehen, die dann effizient für Zwecke der Multicast-Kommunikation adressiert werden können. Hierfür sind die Adressen 128 bis 255 reserviert. Zwei ausgezeichnete Gruppenadressen haben eine besondere Bedeutung: Die Adresse 128 ist als Gruppenadresse für all die Stationen festgelegt, die als redundante Buskopfstationen fungieren können (vgl. Kap. 7.4.2) und die Adresse 255 ist für Mitteilungen an alle Stationen, den sogenannten *Broadcast*, reserviert. Die Adressinformation belegt damit genau ein Byte.

Protokoll- und Dienstdateneinheiten

In DQDFB werden drei verschiedene Typen von Protokolldateneinheiten unterschieden:

- Vorbelegte Übertragungsrahmen (PA-Rahmen) zum periodischen Austausch von Daten,

- Rahmen, die in einem verteilten Warteschlangenverfahren angefordert werden können (QA-Rahmen) und
- Management (MN)-Rahmen, die periodisch ausgetauscht werden.

Alle Rahmen beinhalten das Zugriffskontrollfeld (*Access Control Field, ACF*), das insbesondere auch Informationen über die Rahmenart enthält. Eine Spezifikation dieser Protokolldateneinheiten und der zulässigen Parameterwerte sowie deren Bedeutung ist in Anhang B.1 enthalten.

Um auch größere Dateneinheiten übertragen zu können, wird von DQDFB ein Mechanismus bereitgestellt, der auf der Senderseite eine Segmentierung der Daten und auf der Empfängerseite deren Reassemblierung vornimmt. Durch diesen Mechanismus können Protokolldateneinheiten des Dienstbenutzers mit einer Länge von bis zu 247 Byte übertragen werden. Dazu wird von DQDFB aus der *MSDU* (MAC Service Data Unit) des Dienstbenutzers die *IMPDU* (Initial MAC Protocol Data Unit) durch Hinzufügen der Längeninformation („Payload length") gewonnen.

Jedes Segment einer IMPDU wird in einer *DMPDU* (Derived MAC PDU) übertragen, deren Format in Anhang B.1 dargestellt ist. Eine DMPDU beinhaltet eine Sequenznummer und eine Kennung des Typs der DMPDU. Dieser kann die Werte „SSM" (Single Segment Message) zur Anzeige, daß es sich um eine Nachricht handelt, die in einem Segment übertragen wird, „BOM" (Begin Of Message), „EOM" (End Of Message) zur Anzeige des Beginns bzw. des Endes einer zusammengesetzten Nachricht, oder „COM" (Continuation of Message) zur Anzeige, daß das Segment ein Teil einer zusammengesetzten Nachricht, aber weder deren erstes noch deren letztes Segment ist, annehmen. Die Aufgliederung kurzer, d.h. bis zu 30 Byte langer, als SSM übertragener, und längerer, in mehreren Segmenten übertragener, MSDUs ist in Bild 51 dargestellt.

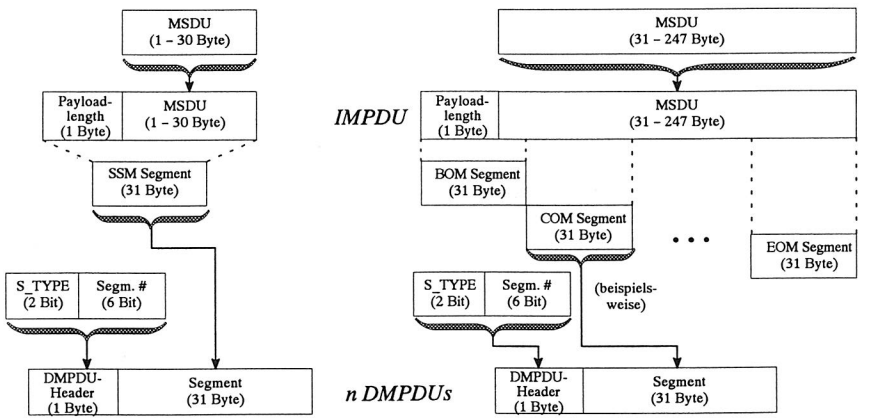


Bild 51: Hierarchie der IM- und DM-Protokolldateneinheiten bei der Übertragung kurzer (links) bzw. längerer Nachrichten (rechts)

PA- und QA-Rahmen dienen der Übertragung von DMPDUs, deren Länge auf 32 Byte festgelegt ist. Für Adress- und rahmenspezifische Informationen sind in PA- und QA-Rahmen jeweils weitere 20 Bit reserviert. Diese Protokollinformationen sind durch eine 4 Bit lange Prüfsumme

(CRC4) mit dem in Formel 7 dargestellten Generatorpolynom geschützt.

$$P_{CRC4}(x) = x^4 + x + 1 \qquad (7)$$

Die Verwendung dieses Polynoms zur Absicherung von 20 Bit Protokollinformation erlaubt die sichere Erkennung aller Fehlersituationen, in denen zwei oder eine beliebige ungerade Anzahl von Bitfehlern auftreten sowie von Fehlersituationen, in denen bis zu drei benachbarte Bits fehlerhaft sind (sog. *burst errors*). Längere *burst errors* werden mit großer Wahrscheinlichkeit erkannt [135]. Bild 52 veranschaulicht ergänzend zu den Darstellungen in Bild 51 die weitere Behandlung von DMPDUs und den Aufbau von QA- und PA-Rahmen. Für das Zugriffsverfahren ist eine Untergliederung der PA- und QA-Rahmen notwendig. Es werden dabei folgende Rahmentypen unterschieden:

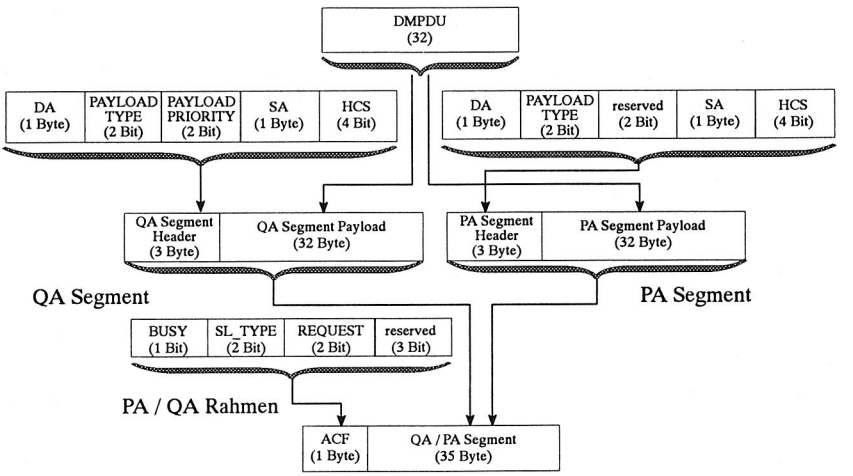


Bild 52: Hierarchie der PA/QA-Protokolldateneinheiten

- *PA/QA-Publish-Request (PA-PR, QA-PR)*: Diese Rahmen dienen der Übertragung von Kennungen von über DRSLP auszutauschenden Variablen. Diese Rahmen werden grundsätzlich an genau eine Station gesendet. Sender eines derartigen Rahmens ist bei PA-PR-Rahmen die BKS und bei QA-PR-Rahmen eine beliebige Station.
- *PA/QA-Publish-Value (PA-PV, QA-PV)*: Diese Rahmen dienen zum Senden des Wertes einer Variablen durch DRSLP. Reservierte PA-PV-Rahmen werden von der BKS erzeugt und an eine Station gesendet, die sie mit Nutzdaten belegt und im Broadcast weitersendet. QA-PV-Rahmen werden grundsätzlich im Broadcast gesendet.
- *PA/QA-Message (PA-M/QA-M)*: Diese Rahmen dienen zum Senden von Nachrichten. Die BKS erzeugt reservierte PA-M-Rahmen und sendet sie an eine Station. Diese nutzt den reservierten Rahmen und sendet ihn an eine beliebige Zieladresse. QA-M-Rahmen können ebenfalls an eine beliebige Zieladresse gesendet werden.
- *QA-Empty (QA-E)*: Diese ungenutzten Rahmen können von einer Station bei entsprechender Zugriffsberechtigung in einen beliebigen QA-Rahmentyp umgewandelt werden. Bei QA-E-Rahmen findet keine Auswertung der Adressinformation statt.

Eine weitergehende Darstellung der Protokolldateneinheiten ist in Anhang B.1 enthalten.

Segmentierung und Reassemblierung

DQDFB stellt Mechanismen zur Verfügung, die eine Übertragung auch von Dateneinheiten erlauben, die größer sind, als das Nutzdatenfeld eines QA- bzw. PA-Übertragungsrahmens. Diese Funktionalität ist für den Dienstbenutzer transparent.

Nach der Übergabe der zu übertragenden Dateneinheit mit einem Dienst zur Anforderung eines periodischen oder aperiodischen Datenaustausches wird die Länge der Nutzdaten überprüft. Liegt sie zwischen 0 und 30 Byte, so können die Daten in einer einzigen DMPDU übertragen werden, der dann der Typ „SSM" zugewiesen wird. Ansonsten werden die Nutzdaten auf der Senderseite in einzelne Segmente zerlegt und in DMPDUs verpackt, denen als Typ „BOM", „COM" oder „EOM" zugewiesen wird. Diese Pakete werden in die Sendewarteschlangen für den periodischen bzw. aperiodischen Nachrichtenaustausch eingereiht.

Das erste Segment einer segmentierten Dateneinheit, beinhaltet 30 Byte Nutzdaten der MSDU und alle nachfolgenden „COM"-Segmente enthalten 31 Byte Nutzdaten der MSDU. Das „EOM"- Segment beinhaltet zwischen 1 und 31 Byte Nutzdaten. Die Anzahl zur Übertragung einer Dateneinheit benötigter Segmente $N_{Seg}(n)$ beträgt in Abhängigkeit von der Länge n der Nutzdaten:

$$N_{Seg}(n) = \left\lceil \frac{(n-30)}{31} \right\rceil + 1 \qquad (8)$$

Die Reassemblierung einer segmentierten Nachricht wird in der Empfängerstation unter der Kontrolle eines Zustandsautomaten (*Reassembly State Machine, RSM*) durchgeführt. Diese kann fehlende Segmente anhand der Sequenznummern erkennen. Eine Fehlerkorrektur ist auf Ebene von DQDFB nicht vorgesehen. Unvollständig übertragene IMPDUs werden verworfen.

Es kann in einer Station durchaus der Fall eintreten, daß gleichzeitig mehrere segmentierte Nachrichten von verschiedenen Stationen empfangen werden. Die Auswahl einer RSM ist anhand der Übertragungsart (periodisch bzw. aperiodisch) und der Adresse der Senderstation eindeutig möglich. Die Funktionsblöcke, die von Dateneinheiten beim Senden bzw. beim Empfang durchlaufen werden, sind in Bild 53 dargestellt.

Schnittstelle zur Bitübertragungsschicht

Zur Übertragung der Nachrichten werden die Dienste der Bitübertragungsschicht genutzt. Das DQDFB-Protokoll setzt dazu die benötigten Parameter korrekt und übergibt sie zusammen mit einem Oktett Nutzdaten an der Schnittstelle zu dieser Schicht.

5.3.2 Funktionelle Architektur von DQDFB-Stationen

DQDFB-Stationen besitzen die in Bild 54 dargestellte funktionelle Architektur. Diese beinhaltet Festlegungen bezüglich der Bitübertragungsschicht und der DQDFB-Schicht. Zur Erbringung der an der Schnittstelle zum LLC-Protokoll angebotenen Dienste nutzt das Protokoll der DQDFB-Schicht eigene Funktionen, solche der Partnerinstanz und solche der Bitübertragungsschicht.

Die DQDFB-Schicht beinhaltet vier Funktionsblöcke. Die allgemeinen Funktionen dienen in regulären Stationen der Weitergabe eintreffender Übertragungsrahmen über den jeweils anderen Dienstzugangspunkt der Bitübertragungsschicht, der Steuerung des Zugriffs auf diese Rahmen, sowie der Aufspaltung eines DQDFB-Übertragungsrahmens in einzelne Oktette auf der Sender- und der Reassemblierung der Rahmen auf der Empfängerseite. Des weiteren beinhalten die allgemeinen Funktionen die Schnittstelle zum Lesen und Schreiben von Rahmeninhalten für die QA-

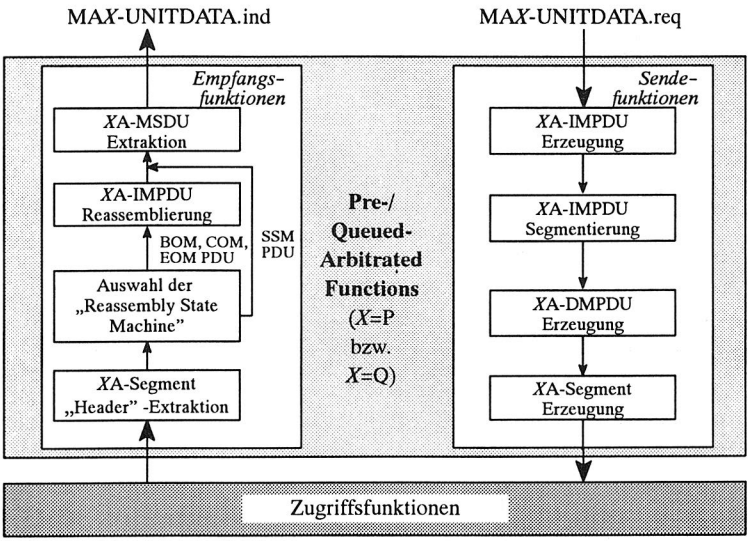


Bild 53: Funktionen beim Senden und beim Empfang von MSDUs

und PA-Funktionen. Die Instanz von DQDFB gehört zu den durch das Systemmanagement verwalteten Objekten und bietet eine entsprechende funktionale Schnittstelle an (vgl. Kapitel 7.1).

Um entgegengenommene Übertragungsanforderungen abwickeln zu können, beinhalten die Buszugriffsfunktionen Verwaltungsmechanismen. Anforderungen bezüglich des hoch- und niederprioren aperiodischen Datenaustausches werden ebenso wie Anforderungen für den periodischen Austausch von Variablen und periodischen Nachrichten in je einer FIFO (First In First Out)-Warteschlange pro Bus zwischengespeichert. In den Warteplätzen befinden sich jeweils vollständige DQDFB-PDUs. Diese Architektur erlaubt es der Instanz von DQDFB, nach Eintreffen eines für die Station reservierten PA-Rahmens bzw. eines QA-Rahmens, auf den die Station Zugriff hat, die zu sendende PDU aus einer Warteschlange zu entnehmen und so die Latenzzeit der Station klein zu halten.

5.3.3 Steuerung des Zugriffs auf Übertragungsrahmen

Grundsätzlich sind in OSIRIS alle Stationen gleichberechtigt und können nach dem gleichen Verfahren auf das Kommunikationsmedium zugreifen. Die Steuerung des Zugriffs auf das Medium bezieht sich auf die Zeitpunkte, zu denen eine Station mit dem Senden eines Übertragungsrahmens beginnt. Diese sind durch die Latenzzeit und den Zeitpunkt des Eintreffens eines Übertragungsrahmens bei der Station fest vorgegeben.

Die eigentliche Problematik besteht in der Steuerung des Zugriffes auf die Übertragungsrahmen zum Zwecke des Sendens von Dateneinheiten, die durch die Station produziert werden. Diese Aufgabe wird von den *Zugriffsfunktionen* einer DQDFB-Station übernommen. Anhand der ersten drei Byte Protokollinformation eines beliebigen DQDFB-Übertragungsrahmens ist diese funktionale Einheit in der Lage zu entscheiden, welche Aktionen ausgeführt werden müssen. Die damit mindestens vor dem Beginn des Sendens abzuwartenden 24 Bitzeiten bestimmen weitgehend die Latenzzeit. Diese verlängert sich noch um die mit t_{Bereit} bezeichnete Zeit, die vergeht, bis der zu sendende

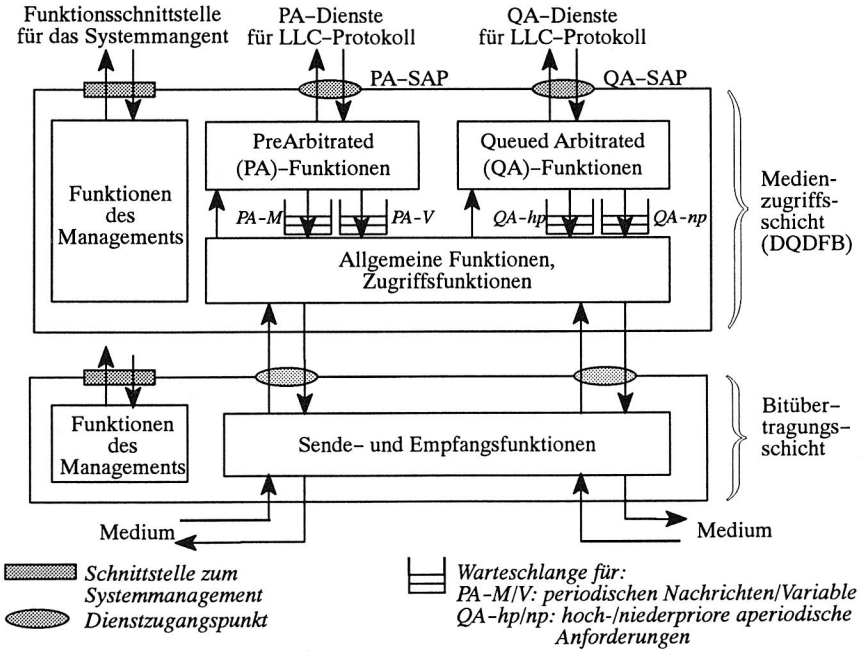


Bild 54: Funktionelle Architektur einer DQDFB-Station

Rahmen lokal ausgewählt und bereitgestellt wurde, so daß sich die Latenzzeit der Station n wie in Formel 9 dargestellt zusammensetzt.

$$t_{Lat}(n) = 24 \times t_{Bit} + t_{Bereit}(n) \qquad (9)$$

Abhängig vom Typ und den Werten ergänzender Protokollverwaltungsinformationen, wie z.B. der Zieladresse und dem Belegungsstatus des empfangenen Rahmes und dem Zustand der lokalen Protokollinstanz wird auf dem gleichen Bus entweder

- der Rahmen unverändert weitergeleitet,
- ein anderer, vorbereiteter Rahmen, der Nutzdaten der Station enthält, gesendet oder
- der eingetroffene Rahmen in einer modifizierten Fassung weitergeleitet.

Die letztgenannte Alternative bezieht sich auf die Behandlung eingetroffener periodisch ausgetauschter Management-Rahmen. Für jeden Rahmen können zudem die Werte des Zugriffskontrollfeldes zur Anforderung von aperiodischer Kommunikation modifiziert werden.

Die vorbereiteten Rahmen werden nach ihrer Erzeugung in FIFO-Warteschlangen zwischengespeichert. Diese Warteschlangen dienen als Schnittstelle zwischen den Zugriffsfunktionen von DQDFB und den PA- bzw. QA-Funktionen (vgl. Bild 54). Dabei existieren insgesamt vier Warteschlangen zur Zwischenspeicherung von periodisch ausgetauschten Variablen bzw. Nachrichten und hoch- bzw. niederprior aperiodisch zu übertragenden Dateneinheiten.

Grundsätzlich werden alle eintreffenden Rahmen, die als Zieladresse eine Gruppenadresse besitzen, abgesehen von einer möglichen Modifikation des Zugriffskontrollfeldes, unverändert weitergeleitet. Die Nutzdaten werden an den Dienstbenutzer übergeben. Bei Rahmen vom Typ PA/QA-PR werden die Nutzdaten des Rahmens an den Dienstbenutzer, die lokale Instanz von DRSLP, übergeben. Der Rahmen selbst wird in einen QA-E-Rahmen umgewandelt und weitergeleitet. Eine derartige Umwandlung ist aufgrund der aktiven Stationsanschaltung durch jede Station möglich und dient durch die Wiederverwendung nicht mehr benötigter, ehemals belegter Rahmen der Erhöhung der Kommunikationsbandbreite.

Trifft ein an die Station adressierter Rahmen vom Typ PA-PV ein, so ist dieser für die Station zum Senden von periodischen Dateneinheiten zum Zwecke der Wahrung der Konsistenz der durch DRSLP verwalteten verteilten Datenbasis reserviert. Die Instanz von DQDFB entnimmt in diesem Fall der FIFO-Warteschlange für periodischen Variablenaustausch die erste Protokolldateneinheit und sendet sie. Entsprechendes geschieht für die erste Protokolldateneinheit in der Warteschlange für periodischen Nachrichtenaustausch, wenn ein an die Station adressierter, freier Rahmen, vom Typ PA-M eintrifft. Liegt in der jeweils betrachteten Warteschlange keine Protokolldateneinheit vor, so wird der Rahmen in einen QA-E-Rahmen umgewandelt und weitergeleitet.

Ein an die Station adressierter, belegter Rahmen vom Typ QA-M oder PA-M enthält Nutzdaten die nur für die lokale Station bestimmt sind. Diese werden an den Dienstbenutzer übergeben. Der Rahmen wird wiederum in einen QA-E-Rahmen umgewandelt und weitergeleitet.

Der Zugriff auf eintreffende QA-E-Rahmen basiert auf dem von DQDB übernommenen Verfahren der verteilten Warteschlange unter Berücksichtigung von zwei Prioritätsstufen, das nachfolgend kurz skizziert wird. Veranschaulichungen dieses Verfahrens sind z.B. im IEEE 802.6 Standard „DQDB" [133] zu finden.

Jede Station verwaltet pro Bus und pro Prioritätsstufe einen *Request (RQ)*- und einen *Countdown (CD)*-Zähler. Der Inhalt des RQ-Zählers zeigt jeweils an, wieviele stromabwärts liegende Stationen bereits einen Sendewunsch dieser oder einer höheren Priorität abgesetzt haben. Dafür ist es erforderlich, daß sowohl die Übertragungsanforderungen der eigenen als auch der höheren Priorität gezählt werden. Dies beinhaltet das Inkrementieren des RQ-Zählers bei Erkennung einer Sendeanforderung gleicher oder höherer Priorität auf dem Gegenbus sowie das Dekrementieren des gleichen Zählers, wenn auf dem Bus ein freier Rahmen passiert. Der Stand des CD-Zählers gibt an, wieviele freie QA-Rahmen die Station passieren lassen muß, bevor sie selbst ihre Daten senden darf.

Zur Anzeige eines Sendewunsches mit einer gewählten Priorität für einen Bus setzt die Station in einem auf dem Gegenbus passierenden Rahmen mit noch ungenutztem Request-Feldes das entsprechende Bit dieses Feldes (vgl. Anhang B.1.1) und kopiert den zum ausgewählten Bus und der benutzten Priorität gehörigen RQ-Zähler in den zugehörigen CD-Zähler. Für jeden auf dem Bus passierenden freien QA-Rahmen der gewählten Priorität wird der zugeordnete CD-Zähler dekrementiert, für jede auf dem Gegenbus während des Wartens passierende höherpriore Anforderung inkrementiert. Erreicht der CD-Zähler den Wert „0", so kann die Station den nächsten freien QA-Rahmen nutzen. Pro Bus und pro Priorität ist dabei jeweils nur eine Anforderung erlaubt. Dadurch wird gewährleistet, daß neue hochpriore Anforderungen Vorzug vor bereits vorhandenen niederprioren, aperiodischen Übertragungswünschen erhalten.

Prinzipbedingt sind bei diesem Verfahren die nahe der Buskopfstation liegenden Stationen benachteiligt, denn sie erhalten statistisch gesehen weniger oft die Möglichkeit, einen aperiodi-

schen Übertragungswunsch abzusetzen. Das zur Erreichung einer diesbezüglich größeren Fairneß vorgesehene, in DQDB optionale, Verfahren des Bandbreitenausgleichs (BWB, BandWidth Balancing) ist Bestandteil des DQDFB-Protokolls. Die für das Verfahren benötigten Werte für die Parameter sind abhängig von der Konfiguration des Systems im Einzelfall zu setzen. Eine genaue Beschreibung des Verfahrens findet sich wiederum u.a. im IEEE 802.6 Standard.

5.3.4 Dienste der zugriffsteuernden Teilschicht

Das DQDFB-Protokoll stellt an seiner Diensteschnittstelle folgende Dienste zur Verfügung, deren Namen je nachdem, ob sie allgemeiner Ausrichtung sind oder dem periodischen bzw. dem aperiodischen Austausch von Dateneinheiten dienen, das Präfix „MA" (Medium Access), „MAP" (MA-Periodic) bzw. „MAQ" (MA-Queued) vorangestellt ist:

- `MAP-UNITDATA`: zum Austausch von Dateneinheiten mittels der periodischen Datenübertragungsrahmen von DQDFB.
- `MAQ-UNITDATA`: zum aperiodischen Austausch von Dateneinheiten mittels des Verfahrens der verteilten Warteschlange.
- `MA-SENT`: Lokale Bestätigung zur Anzeige des erfolgten Sendens einer periodisch oder aperiodisch zu übertragenden Dateneinheit.

Diese Dienste werden an dem jeweiligen Dienstzugangspunkten, dem QA-SAP für den aperiodischen und dem PA-SAP für den periodischen Datenaustausch, angeboten.

MAP-UNITDATA

Der Dienst MAP-UNITDATA dient zur Übergabe von Dateneinheiten, die mittels des periodischen Datenaustauschmechanismus übertragen werden sollen bzw. wurden.

Mit Hilfe des Dienstelementes `MAP-UNITDATA.ind` wird der Protokollinstanz von DRSLP angezeigt, daß ein an sie adressierter, von der Station mit der Adresse „Source Address" gesendeter, PA-Rahmen empfangen wurde, der Nutzdaten enthält. Mittels des Parameters „Class" wird differenziert, ob es sich um die Nutzdaten eines PA-PR-, PA-PV- oder PA-M-Rahmens handelt. Das Nutzdatum für die Instanz von DRSLP ist im Parameter „Data" enthalten. Entsprechendes gilt für das Dienstelement der Anforderung, wobei hier nur die Wahl zwischen dem Senden eines PA-PV- oder eines PA-M-Rahmens besteht. Dabei wird der Empfänger der Nachrichten als Wert des Parameters „Destination Address" übergeben. Die bei der Dienstanforderung zu besetzende Aufrufkennung („Request-Identifier") wird für die Zuordnung einer `MA-SENT`-Dienstankündigung zur einer Sendeanforderung benötigt.

Parametername	req	ind
Argument	M	M
Source Address		M
Destination Address	M	
Class	M	M
Data	M	M
Request-Identifier	M	

Tabelle 7: Parameter der Dienstelemente des Dienstes `MAP-UNITDATA`

Durch die Instanz von DQDFB wird aus den mittels der Dienstanforderung übergebenen Daten und lokal vorliegenden Informationen, z.B. der eigenen Stationsadresse, ein entsprechender PA-Rahmen zusammengesetzt und - je nach der gewählten Klasse - in die entsprechende Warte-

schlange eingereiht. Nachrichten, die nur an eine andere Station gesendet werden sollen, werden nur auf dem Bus gesendet, über den die Zielstation erreicht werden kann. Die Information darüber, welcher Bus auszuwählen ist, kann vom Systemmanagement lokal erfragt werden. PA-PV und PA-M-Rahmen, die im Multicast gesendet werden sollen, werden grundsätzlich über beide Busse gesendet. Dabei kann es in Abhängigkeit der Lage der Station zur Buskopfstation dazu kommen, daß die zusammengehörenden Rahmen, z.B. zwei PA-PV-Rahmen, die auf den beiden Bussen gesendet werden, zu unterschiedlichen Zeitpunkten bei der Station eintreffen. Wird eine Nachricht, die nur an eine einzelne Station adressiert ist, mittels der periodischen Nachrichtenübertragungsdienste übertragen, so wird der zugehörige periodische Nachrichtenübertragungsrahmen auf dem Gegenbus in einen QA-E-Rahmen umgewandelt.

MAQ-UNITDATA

Der Dienst MAQ-UNITDATA dient zur Anforderung des Sendens einer Dateneinheit im aperiodischen Modus bzw. zur Ankündigung, daß eine für die Station bestimmte, im aperiodischen Modus ausgetauschte, Dateneinheit eingetroffen ist.

Die Parameter (vgl. Tab. 8) und ihre Bedeutung entsprechen weitgehend denen, die bereits bei dem Dienst MAP-UNITDATA eingeführt wurden. Ergänzend dazu kann der Dienstbenutzer des Dienstes MAQ-UNITDATA spezifizieren, ob der Dienst mit hoher oder niedriger Priorität ausgeführt werden soll.

Parametername	req	ind
Argument	**M**	**M**
Source Address		M
Destination Address	M	
Priority	M	M
Class	M	M
Data	M	M
Request-Identifier	M	

Tabelle 8: Parameter der Dienstelemente des Dienstes `MAQ-UNITDATA`

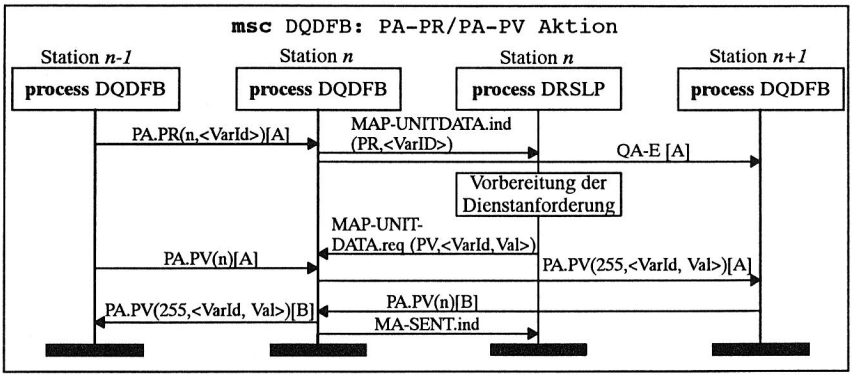
Die Wahl eines Wertes für diese Dienstqualität wirkt sich auf die Abarbeitungsreihenfolge der Anforderungen aus. Die Anforderungen werden von den QA-Funktionen von DQDFB in die entsprechende Warteschlange eingereiht. Solange in den Warteschlangen für den aperiodischen Nachrichtenaustausch Anforderungen vorliegen, fordern die Zugriffsfunktionen von DQDFB über den Mechanismus der verteilten Warteschlange QA-Rahmen für die Station an. Der Überlauf einer Warteschlange führt zur einer Fehlermeldung an das Systemmanagement.

MA-SENT

Sowohl für periodisch als auch für aperiodisch zu sendende Dateneinheiten wird der Sendevorgang überwacht. Der lokale Dienst `MA-SENT` dient zur Benachrichtigung des Dienstbenutzers über den Vollzug des Sendens einer Dateneinheit. Für diesen Dienst ist nur das Dienstelement der Ankündigung definiert. Als einziger Parameter wird die Aufrufkennung der Sendeanforderung für die gesendete Dateneinheit übergeben.

Die Protokollabläufe beim periodischen und aperiodischen Austausch von Variablen sind in den Bildern 55 und 56 in Form von Nachrichten-Abfolge-Diagrammen dargestellt. Dabei ist das Verhalten von DRSLP als Dienstbenutzer bereits berücksichtigt.

Die Reihenfolge des Eintreffens der reservierten Rahmen in Bild 55 und die des gewährten Zugriffs auf leere QA-Rahmen in Bild 56 ist beliebig und wurde ohne Beschränkung der Allge-



PA.PR(n,<VarId>)[X]: *PA-Rahmen, Type: Publish-Request für Station „n"; bezogene Variable wird durch das Nutzdatum „VarId" bezeichnet; Übertragung auf Bus A*

QA-E[A]: *Freier QA-Rahmen, der auf Bus A übertragen wird*

MAP-UNITDATA.ind(PR,<VarId>): Dienstankündigung, Typ: Publish-Request; für Variable „VarId"

MAP-UNITDATA.req(PV,<Var>): Dienstanforderung, Typ: Publish-Value; Var.: „VarId", Wert: „Val"

PA-PV(n)[X]: *Für Station n reservierter PA-PV-Rahmen, der auf Bus X übertragen wird*

PA-PV(255,<VarId, Val>)[X]: Belegter PA-PV-Rahmen, der auf Bus X übertragen wird

MA-SENT.ind: *Dienstankündigung: Variable gesendet*

Bild 55: Austausch von Variablen in periodischen Übertragungsrahmen

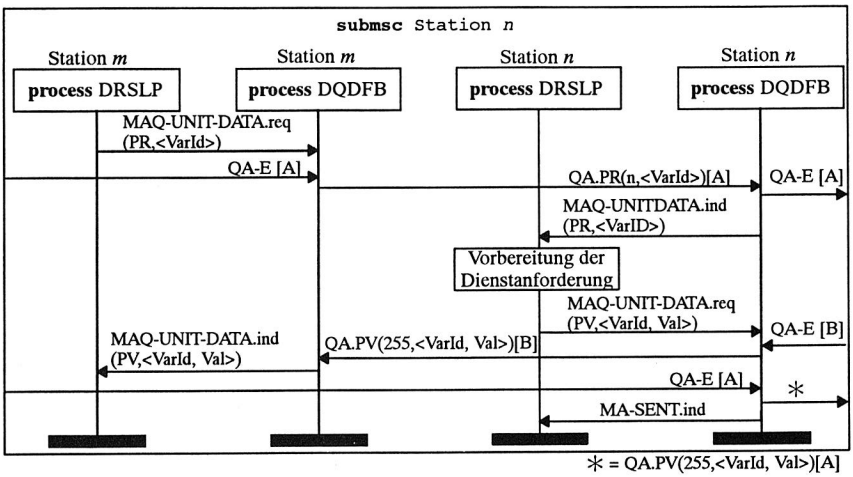
meingültigkeit jeweils beispielhaft gewählt. Die Beschreibung der einzelnen Nachrichten erfolgt nur bei deren erstem Auftreten.

5.3.5 Funktionalität und Rolle der Buskopfstation

Die BKS beinhaltet neben der Funktionalität einer regulären Station zusätzlich Mechanismen zur Erzeugung und zur Vernichtung von Übertragungsrahmen. Die Erzeugung der Rahmen erfolgt in einem festen Takt, der durch die Übertragungsgeschwindigkeit, die Übertragungsart und die Rahmenlänge bestimmt wird.

Der Typ des nächsten, für einen Bus zu erzeugenden, Rahmens wird aus der zugeordneten Pollingtabelle entnommen. Die Pollingtabellen für die Busse A und B unterscheiden sich nur dort, wo periodische „Publish-Request" Anforderungen gesendet werden. Diese werden grundsätzlich nur über einen Bus gesendet. Die Wahl des Busses wird während der Konfigurierung des Netzwerkes vorgenommen. Sie erfolgt so, daß die Publish-Request Anforderung vor dem ersten zugehörigen, reservierten „Publish-Value" Übertragungsrahmen auf dem entgegengesetzt orientierten Bus bei der Station eintrifft.

Beide Pollingtabellen werden zyklisch abgearbeitet und können zur Laufzeit dynamisch durch das Systemmanagement modifiziert werden. Die von den Anforderungen an die Periodizität der Übertragung von Variablen und Nachrichten abhängige Koinzidenz der Übertragungszeitpunkte bestimmt die Länge der Pollingtabellen. Sie beinhalteten genau so viele Einträge, wie Rahmen in der Zeit gesendet werden müssen, die das kleinste gemeinsame Vielfache der Periodizitäten der periodischen Austauschwünsche unter Berücksichtigung der Anzahl der pro Anforderung zu sendenden Rahmen darstellt. In Abhängigkeit von der Rahmenzeit und den Periodizitäten der Über-



MAQ-UNITDATA.req(PR,<VarId>): *Dienstanforderung, Typ: Publish-Request; Variable: „VarId"*

QA.PR(n,<VarId>)[A]: *QA-Rahmen, Typ: Publish-Request für Station n; Variable: „VarId" Übertragung auf Bus A*

MAQ-UNITDATA.ind(PR,<Var>): *Dienstanforderung, Typ: Publish-Request; Variable: „VarId"*

MAQ-UNITDATA.req(PV,<Var, Val>): *Dienstanforderung, Typ: Publish-Value; Variable: „VarId", Wert: „Val"*

QA-PV(255,<VarId, Val>)[X]: *QA-PV-Rahmen, der auf Bus X übertragen wird*

MAQ-UNITDATA.ind(PV,<Var, Val>): *Dienstankündigung, Typ: Publish-Value; Variable „VarId" Wert: „Val"*

Bild 56: Aperiodischer Austausch von Variablen

tragungswünsche ist zudem ein Jitter zu berücksichtigen, der allerdings durch gezieltes Einstreuen eines zusätzlichen QA-Rahmens auf maximal eine Rahmenzeit begrenzt werden kann.

Die Pollingtabelle beinhaltet Informationen über die Art des jeweils zu erzeugenden Übertragungsrahmens und gegebenenfalls über die benötigten Zusatzinformationen, wie z.B. die Adresse der Station, an die ein PA-Rahmen zu senden ist und die Identifikation der von dieser Station erzeugten Variablen, die über den Bus ausgetauscht werden soll. Ein Eintrag in die Pollingtabelle hat die in Bild 57 in der Syntaxnotation ASN.1 dargestellte Struktur.

```
POLLING_LIST_ENTRY ::= SEQUENCE {
      id              INTEGER,
      typ             SLOT_TYPE,
      mode            MODE_TYPE OPTIONAL,
      destination     ADDRESS OPTIONAL,
      var_id          VARIABLE-ID OPTIONAL
}
```

Bild 57: Darstellung der Datenstruktur eines Eintrages in der Pollingtabelle

Der Parameters „id" erlaubt eine eindeutige Referenzierung eines Eintrages in der Pollingtabelle. Der Typ des zu erzeugenden Rahmens, QA- oder PA-Rahmen, ist im Parameter „typ" enthalten.

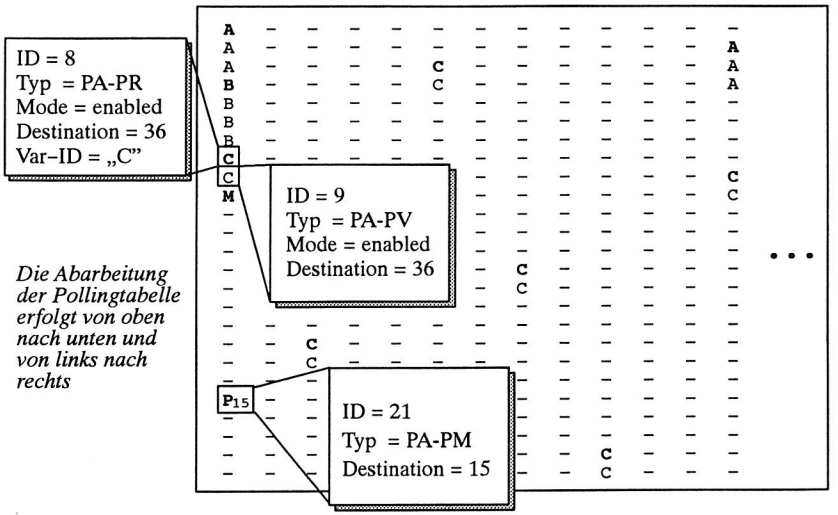


Bild 58: Ausschnitt aus der Pollingtabelle mit Einträgen für Publish-Request, Publish-Value und Management-Rahmen

Bei PA-Rahmen erfolgt hier zusätzlich eine Spezifikation des Typs (Publish-Request bzw. Publish-Value). Zudem besteht für PA-Rahmen die Möglichkeit, daß der Eintrag gesperrt ist. In diesem Fall hat der Parameter „mode" den Wert „disabled", ansonsten den Wert „enabled" und es wird anstelle des PA-Rahmens ein QA-Rahmen gesendet. Vorbelegte Rahmen müssen mit der Adresse der Zielstation versehen werden. Die benötigte Information ist im Parameter „destination" enthalten. Für PA-Rahmen, die zur Ankündigung des Austausches von Variablen genutzt werden, ist in dem Parameter „var-id" die Kennung der zu übertragenden Variablen enthalten. Die nachfolgend an die Station gesendeten PA-Rahmen vom Typ „Publish-Value" sind den Anforderungen implizit zugeordnet. Der beispielhafte Aufbau einer Pollingtabelle mit mehreren Einträgen zur periodischen Übertragung von Daten ist in Bild 58 gezeigt. In dem dargestellten Auszug aus der Pollingtabelle werden zur Verdeutlichung Buchstaben zur Kennzeichnung von Variablen und das Symbol „–" zur Darstellung von zu sendenden QA-Rahmen verwendet. Variable „A" benötigt dabei zwei Rahmen und wird periodisch alle 50 ms ausgetauscht. Die Nutzdaten von Variable „B" werden unter Nutzung von drei aufeinanderfolgenden Rahmen alle 100 ms übertragen. Variable „C" benötigt nur einen Rahmen und wird alle 10 ms ausgetauscht. Die Publish-Request-Rahmen sind dabei in Fettschrift dargestellt. Zur periodischen Übertragung von Managementinformationen wird alle 250 ms ein diesbezüglicher, mit „M" gekennzeichneter, Rahmen übertragen. Der Übertragungsrahmen für periodische Nachrichten der Station mit der Nummer 15 ist mit „P_{15}" gekennzeichnet. Die diesbezüglich festgelegte Periodizität beträgt 300 ms. Zwischen zwei aufeinanderfolgenden Rahmen wird eine Ruhepause von 24 Bitzeiten eingehalten.

5.4 Funktionalität und Protokoll der LLC-Teilschicht

Die wesentliche Aufgabe des mit DRSLP bezeichneten, in dieser Arbeit ebenfalls neu entwickelten, Protokolls der LLC-Teilschicht der Sicherungsschicht von OSIRIS besteht darin, die Übertra-

gung von verschiedenen Klassen von Daten entsprechend deren besonderer Bedürfnisse auf Basis der von DQDFB bereitgestellten Dienste durchzuführen. Gemäß der Anforderungsdefinition besteht die Notwendigkeit,

- Prozeßdaten periodisch und aperiodisch auszutauschen,
- die Übertragung kontinuierlicher und diskreter Medien zu unterstützen und
- Nachrichten mit und ohne Berücksichtigung von Echtzeitanforderungen versenden zu können.

Zudem obliegt es DRSLP, die für den Prozeßdatenaustausch notwendige verteilte Datenbasis zu verwalten.

5.4.1 Verwendete Grundprinzipien zur Erbringung der Funktionalität

Die Prinzipien, nach denen DRSLP diese Anforderungen erfüllt, die dafür vorgesehenen Protokollmechanismen und die angebotenen Dienste werden in den nachfolgenden Abschnitten vorgestellt.

Verteilte Datenbasis

Insbesondere in industriellen Steuerungs- und Regelungsapplikationen wird häufig eine kurze Zugriffszeit auf die Werte von Prozeßvariablen gefordert. Dies kann durch eine verteilte Kopie der relevanten Informationen (Replikation) erreicht werden, da bei stationsinterner Verfügbarkeit aktueller Werte ein schneller lokaler Zugriff auf die Kopie der Daten stattfinden kann. Die Abwicklung des Replikationsmechanismus, der die Konsistenz der verteilten Kopien der Prozeßvariablen sicherzustellen hat, obliegt dem Kommunikationssystem. Aktuelle Feldbussysteme, wie etwa FIP und ansatzweise PROFIBUS (vgl. Kap. 3.4), machen ebenfalls von diesem Prinzip Gebrauch. Generell wird in diesem Ansatz zwischen *Produzenten* und *Konsumenten* von Variablen unterschieden. Als *Produzent* einer Variablen wird die Station bezeichnet, die deren Wert erzeugt. Als *Konsumenten* werden die Stationen bezeichnet, die eine Kopie der Variablen halten (vgl. Bild 59). Die Festlegung des Rollenverhaltens erfolgt während der Konfigurierung des Netzwerkes.

Im Konzept von OSIRIS wird gemäß der Anforderungen zwischen regulären Prozeßvariablen und Echtzeitvariablen unterschieden. Echtzeitvariable besitzen einen Zeitstempel, aus dem auch die Dauer ihrer Gültigkeit hervorgeht. Damit kann auf weitere diesbezügliche Konsistenzattribute, wie sie z.B. in FIP zur Anzeige der rechtzeitigen Erzeugung und der rechtzeitigen Verteilung von Variablen vorgesehen sind, verzichtet werden.

Ob die Kopien einer Variablen aperiodisch oder periodisch aktualisiert werden müssen, hängt von der Charakteristik des durch sie modellierten Prozeßwertes ab. Diese ändert sich zur Laufzeit des Systems nicht und ist durch den Benutzer des Systems bei der Konfigurierung zu bestimmen. Gleiches gilt für die Periodizität, mit der Variable ggf. ausgetauscht werden.

Aufbauend auf den von DQDFB angebotenen Diensten wird die Verwaltung der verteilten Datenbank in OSIRIS durch die verteilten Instanzen von DRSLP vorgenommen. Der periodische Austausch von Variablen wird unter Kontrolle der Buskopfstation vorgenommen, in deren Pollingtabelle bei der Konfigurierung des Systems entsprechende Einträge vorgenommen werden. Dabei ist auch die Anzahl der zum Austausch von größeren Nachrichten notwendigen Anzahl von PA-Rahmen berücksichtigt. Die DRSLP-Instanz einer Station reagiert nur auf die entsprechenden Dienstankündigungen, d.h. sie stellt nach dem Eintreffen einer Anforderung bezüglich des Austausches einer durch sie produzierten Variablen eine entsprechende Dienstdateneinheit zusammen und übergibt sie mittels des `MAP-UNITDATA`-Dienstes an die Instanz von DQDFB (vgl. Bild 55).

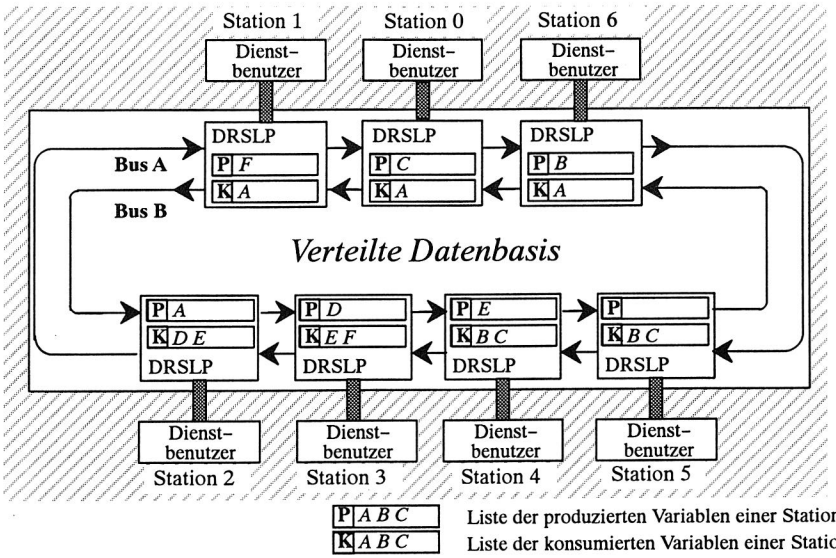


Bild 59: Rollenverteilung „Produzent / Konsumenten" und Transparenz der verteilten Datenbasis

Der Dienstbenutzer von DRSLP ist nicht am periodischen Variablenaustausch beteiligt. Der aperiodische Austausch von Variabalen wird von DRSLP ebenfalls angeboten. Er wird nach expliziter Anforderung des Dienstbenutzers unter Nutzung der Dienste von DQDFB durchgeführt. Um die Konsistenz der verteilten Kopien wahren zu können, werden alle Aktualisierungen von Variablen grundsätzlich im Broadcast durchgeführt.

Um ihrer Aufgabe gerecht werden zu können, hat die DRSLP-Instanz einer jeden Station Zugriff auf die Liste von Variablenbezeichnern, bezüglich derer die Station als Produzent bzw. als Konsument auftritt, sowie auf deren Spezifikation, insbesondere den Speicherplatzbedarf. Die eindeutige Referenzierung einer Variablen wird durch die Variablenadresse ermöglicht. Diese besteht aus zwei Byte. Das höherwertige Byte beinhaltet die Adresse der produzierenden Station, das niederwertige Byte die Adresse der Variablen innerhalb der Station. Damit sind pro Station theoretisch bis zu 256 produzierte Variable für Zwecke der Anwendungen nutzbar. Der Adressbereich von 240 bis 255 ist dabei für die Pufferung von Daten reserviert, die im Rahmen des Autausches von produzierten, kontinuierlichen Multimedia-Datenströmen übertragen werden. Ein aperiodischer Zugriff auf diese Daten ist nicht zulässig. Für den regulären Produktivdatenaustausch sind somit pro Station 240 durch sie produzierte und beliebig viele konsumierte Variable verfügbar. Der Typ der Variablen ist anwendungsspezifisch und auf der Ebene von DRSLP nicht relevant.

Übertragung von Multimedia-Datenströmen

Für den isochronen Austausch von Dateneinheiten, die zu kontinuierlichen multimedialen Datenströmen gehören, werden ebenfalls die Mechanismen des periodischen Datenaustausches im Zusammenspiel zwischen DQDFB, der Buskopfstation und der Instanz von DRSLP genutzt. Im

Unterschied zum periodischen Austausch von Variablen steht bei der Konfigurierung des Systems in der Regel nicht fest, zu welchem Zeitpunkt welcher Datenstrom genutzt wird und welche Bandbreite für ihn benötigt wird. Die Bereitstellung der zur Laufzeit des Systems benötigten Bandbreite gehört zu den Aufgaben des Systemmanagements (vgl. Kap. 7.5.3).

In Analogie zur Unterscheidung von Produzenten und Konsumenten bei Prozeßvariablen wird bei Multimedia-Datenströmen zwischen *Quellen* und *Senken* von Datenströmen unterschieden. Die von einer Quelle erzeugten Dateneinheiten werden in der Quellstation in einem Pufferplatz bis zu ihrer Übertragung zwischengespeichert und in den Senken von DRSLP ohne weitere Speicherung direkt an den Dienstbenutzer übergeben. Die Anforderung ausreichender Bandbreite obliegt der Verantwortung der Applikation ebenso, wie der Start und die Beendigung der Übertragung. Das Senden der zu einem Multimedia-Datenstrom gehörigen Dateneinheiten erfolgt grundsätzlich im Broadcast und ist in Bild 60 beispielhaft für einen isochronen Datenstrom, der durch die Kodierung schwach gleichmäßig wird, dargestellt. Eine *Verarbeitungseinheit V* repräsentiert dabei einen zusammenhängenden Bereich eines Datenstromes, der effizient kodiert und dekodiert werden kann und der eine vom jeweiligen Verfahren abhängige Struktur und Länge besitzt. Vorschläge für eine Behandlung des von einem H.261-Koder erzeugten Datenstromes werden beispielsweise von *Turletti et al.* [136] gemacht.

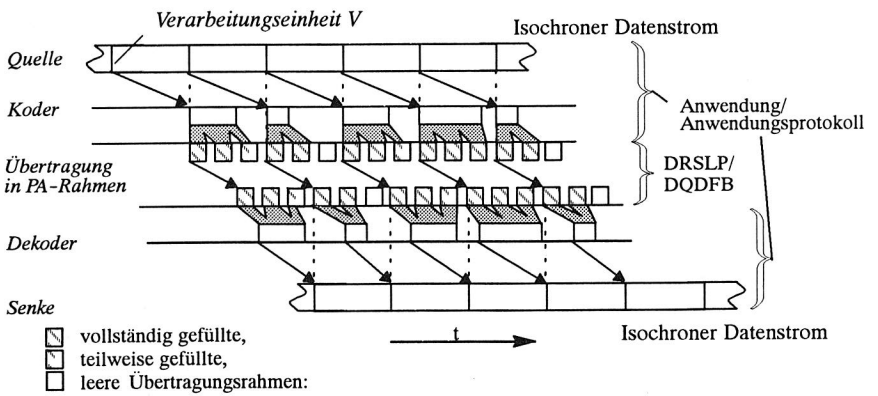


Bild 60: Übertragung von isochronen Multimedia-Daten mit Hilfe des periodischen Datenaustausches

Nachrichtenaustausch

Um den aufgestellten Anforderungen entsprechen zu können, werden durch DRSLP verschiedene Dienstqualitäten für den Austausch von Nachrichten angeboten. Es wird zwischen gesichertem und ungesichertem Nachrichtenaustausch unterschieden. Im Falle des ungesicherten Nachrichtenaustausches wird weiter zwischen solchem, bei dem der Sendevorgang keiner zeitlichen Beschränkung unterliegt, und solchem, bei dem Echtzeitanforderungen berücksichtigt werden müssen, differenziert.

Das ungesicherte Senden erlaubt die Übertragung von Daten an eine beliebige Station bzw. Gruppe von Stationen. Der gesicherte Nachrichtenaustausch findet dagegen zwischen genau zwei

Stationen statt. Bei ungesicherter Nachrichtenübertragung kann durch den Dienstbenutzer eine Zielzeit für das Senden der Nachricht angegeben werden. Mit der Übernahme der Dienstanforderung garantiert DRSLP das Senden der Nachricht bis zu diesem Zeitpunkt. Ist dies nicht möglich, so wird die Dienstanforderung zurückgewiesen. Die Dienst- und Protokollmechanismen, die dem Nachrichtenaustausch zugrundeliegen, werden in Kapitel 5.4.3 näher betrachtet.

Protokolldateneinheiten

Für DRSLP sind Protokolldateneinheiten zur Anforderung des Austausches von Variablen (*Publish-Request-PDU*), zur Übertragung von Variablen (*Publish-Value-PDU*), für den Austausch von zu kontinuierlichen Multimedia-Datenströmen gehörigen Dateneinheiten (*MM-Exchange-PDU*) und zur Kommunikation mittels bestätigter (*ACK-MSG-PDU*) und unbestätigter (*MSG-PDU*) Nachrichten definiert. Die Formate der Protokolldateneinheiten sind in Anhang B.1.2 dargestellt. Zur Erkennung von Übertragungsfehlern in den Nutzdaten dient eine 8-Bit CRC-Prüfsumme, die mit dem in Formel 10 aufgeführten Polynom berechnet wird.

$$P_{CRC8}(x) = x^8 + x^2 + x + 1 \qquad (10)$$

Die Verwendung dieses Polynoms erlaubt die Erkennung aller Fehlersituationen, in denen zwei oder eine beliebige ungerade Anzahl von Bitfehlern auftreten. Außerdem werden *burst errors* mit einer maximalen Länge von sieben benachbarten Bits grundsätzlich, solche mit größerer Länge mit großer Wahrscheinlichkeit erkannt. Alle als fehlerhaft erkannten Protokolldateneinheiten werden verworfen.

5.4.2 Struktur und Schnittstellen von DRSLP

DRSLP nutzt die Dienste von DQDFB an den von DQDFB zur Verfügung gestellten Dienstzugangspunkten für den periodischen und den aperiodischen Datenaustausch. Für jede der von DRSLP erbrachten Datenübertragungsarten stehen Dienstzugangspunkte zur Verfügung. Dabei existiert für die Übertragung von Variablen und Multimedia-Informationen je genau ein Dienstzugangspunkt (DL_VAR_SAP bzw. DL_MM_SAP), der von allen Dienstbenutzern gemeinsam genutzt wird. Zum Austausch von Nachrichten sind 16 Dienstzugangspunkte (DL_MSG_SAP) vorgesehen. Von letzteren ist der DL_MSG_SAP mit der Kennung „0" für den Austausch von Managementinformationen reserviert und der SAP mit der Kennung „15" dient als Default-SAP, über den Nachrichten zugestellt werden, die im Multicast übertragen wurden. Ein Senden von Nachrichten über den Default-SAP ist nicht zulässig. Die SAPs mit den Kennungen „1" bis „14" sind funktional gleichwertig.

Eine Protokollinstanz von DRSLP hat die in Bild 61 vorgestellte Struktur. Die Auswertung der am PA-SAP und QA-SAP von DQDFB übergebenen Dienstdateneinheiten obliegt der jeweils zuständigen Funktionsverwaltung, die entsprechende Aktionen des DRSLP-Protokolls anstößt. Die Funktionsverwaltung für den periodischen Datenaustausch ist dabei für die korrekte Abwicklung der zur periodischen Aktualisierung der verteilten Datenbasis notwendigen lokalen Aktionen zuständig. Sie übergibt dazu eintreffende, aktualisierte Werte von Variablen, die als verteilte Kopie gehalten werden, an die Speicherverwaltung. Zudem stellt sie nach Eintreffen einer Sendeanforderung bezüglich einer produzierten Variable die entsprechende Protokolldateneinheit zusammen und ruft den Dienst `MAP-UNITDATA` auf (vgl. Bild 55). Sie sorgt auch für eine Vorbereitung der Protokolldateneinheiten für den Austausch von Nachrichten, die in für die Station reservierten periodischen Nachrichtenübertragungsrahmen übertragen werden sollen. Eintreffende Bestand-

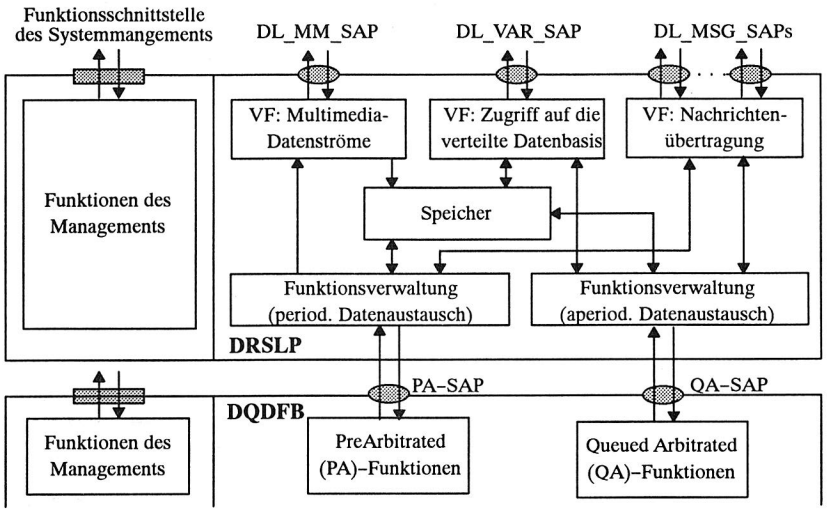


Bild 61: Struktur von DRSLP und Schnittstelle zu DQDFB

teile eines Multimedia-Datenstromes werden direkt an die Funktionseinheit zur Verwaltung des Dienstzugangspunktes weitergeleitet.

Die Funktionsverwaltung für den aperiodischen Datenaustausch nimmt aperiodische Anforderungen bezüglich des Austausches von Variablen entgegen und wickelt sie ab. Diese können dabei durch eine andere Station oder durch den lokalen Dienstbenutzer von DRSLP erzeugt werden. Die Werte der produzierten Variablen, die der Kopien der konsumierten Variablen und die zur Übertragung anstehenden Dateneinheiten aus Multimedia-Datenströmen werden im Speicher von DRSLP gehalten.

Auf diesen greifen auch die Verwaltungsfunktionen (VF) des DL_MM_SAP und des DL_VAR_SAP beim lokalen Schreiben bzw. beim lokalen Schreiben und Lesen von Daten zu. Für Lese- und Schreibanforderungen auf Variable, die nicht lokal abgewickelt werden können, werden die Funktionen des aperiodischen Datenaustausches benutzt.

5.4.3 Dienste zur Nutzung der Funktionalität

DRSLP bietet an den verschiedenen Arten von Dienstzugangspunkten unterschiedliche Dienste an, die im folgenden kurz vorgestellt werden.

Zugriff auf Variablen

Am Dienstzugangspunkt für den Zugriff auf die verteilte Datenbasis stehen Dienste zum lokalen und entfernten Lesen (`DL-READ`, `DL-UPDATE`) und Schreiben (`DL-WRITE`, `DL-PUBLISH`) von Variablen zur Verfügung. Ergänzend sind lokale Dienste definiert, die das Eintreffen eines aktualisierten Wertes einer Variablen bzw. das erfolgte Senden einer lokal produzierten Variable anzeigen.

Zum lokalen Lesen eines Variablenwertes steht der Dienst `DL-READ` zur Verfügung. Der einzige Parameter der Dienstanforderung ist die Identifikation der zu lesenden Variablen. In der lokalen

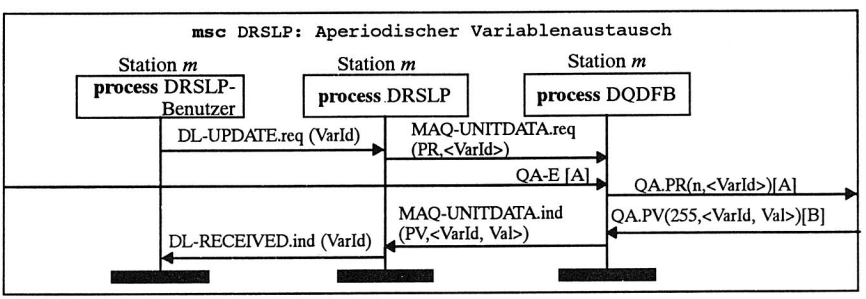
Antwort, die unverzüglich an den Dienstbenutzer übergeben wird, ist neben dieser Identifikation auch der aktuelle Wert der Variablen enthalten. Entsprechendes gilt für den lokalen Schreibdienst `DL-WRITE`, bei dem mit der Anforderung die Identifikation der bezogenen Variable und deren neuer Wert übergeben werden. Das Schreiben von Werten ist dabei nur für solche Variablen zulässig, die von der lokalen Station produziert werden. Für das Lesen bestehen keine diesbezüglichen Beschränkungen.

Um den aperiodischen Austausch des Wertes einer Variablen zu initiieren, die von einer anderen als der lokalen Station produziert wird, steht der unbestätigte Dienst `DL-UPDATE` zur Verfügung. Die Parameter des Dienstelementes zur Anforderung sind in Tabelle 9 aufgeführt.

Parametername	req
Argument	**M**
Variable-ID	M
Priority	M

Tabelle 9: Parameter der Dienstelemente des Dienstes `DL-UPDATE`

Die für die Übertragung zu benutzende Dienstpriorität ist im Parameter „Priority" hinterlegt. Die Dienstanforderung wird auf den Dienst `MAQ-UNITDATA` mit Diensttyp „Publish-Request" abgebildet. In Ergänzung zu der Darstellung in Bild 56 zeigt Bild 62 das Verhalten einer DRSLP-In-



DL-UPDATE.req(VarId): Dienstanforderung: Entferntes Lesen der Variable mit der Kennung „VarId"
DL-RECEIVED.ind(VarId): Dienstankündigung: Variable mit der Kennung „VarId" wurde aktualisiert.

Bild 62: Aperiodischer Austausch von Variablen

stanz auf die Entgegennahme einer `DL-UPDATE`-Dienstanforderung, wobei ohne Beschränkung der Allgemeingültigkeit die Station *m* über Bus A zu erreichen sei. Als verbindendes Element zwischen beiden Bildern dient gemäß der Darstellungskonventionen für *msc* die Instanz „Station *m* ***(process DQDFB)***". Da die `DL-UPDATE`-Dienstanforderung bei der Zielstation durch die Funktionsverwaltung für den aperiodischen Datenaustausch bearbeitet wird, kann auf eine Definition des Dienstelementes der Ankündigung verzichtet werden. Entsprechendes gilt auch für den Dienst `DL-PUBLISH`, mit dessen Hilfe eine Applikation die aperiodische Aktualisierung einer von der lokalen Station produzierten Variablen explizit anstoßen kann. Die Parameter des Dienstelementes enthalten zusätzlich zu denen des Dienstes `DL-UPDATE` noch den Wert der Variablen. Diese Dienstanforderung wird ebenfalls auf den Dienst `MAQ-UNITDATA` von DQDFB abgebildet, wobei für den Diensttyp der Wert „Publish-Value" gewählt wird.

Mittels der Dienstankündigung `DL-RECEIVED.ind` kann die DRSLP-Instanz dem lokalen Dienstbenutzer anzeigen, daß eine von der lokalen Station konsumierte Variable über das Kommunikationssystem aktualisiert wurde. Der einzige Dienstparameter der Dienstankündigung ist die im Parameter `ID` enthaltenen Identifikation der Variablen. Sie wird von DRSLP nach dem Eintreffen einer `MAP-UNITDATA.ind`- bzw. einer `MAQ-UNITDATA.ind`-Dienstankündigung mit Diensttyp „Publish-Value" über den DL_VAR_SAP an den Dienstbenutzer übergeben. Auf den aktuellen Wert der Variablen kann dieser mittels des lokalen Lesedienstes zugreifen.

Wurde der Wert einer Variablen, die von der lokalen Station produziert wird, ausgetauscht, so wird dies dem Dienstbenutzer von DRSLP mittels der Dienstankündigung `DL-SENT.ind` unter Angabe der Identifikation der Variablen mitgeteilt.

Austausch von zu Multimedia-Datenströmen gehörenden Dateneinheiten

Für den Austausch von zu Multimedia-Datenströmen gehörenden Dateneinheiten wird durch DRSLP ein Dienst am entsprechenden Dienstzugangspunkt, dem DL_MM_SAP, bereitgestellt. Die Parameter der Dienstelemente dieses mit „`DL-MM-EXCHANGE`" bezeichneten, unbestätigten Dienstes sind in Tabelle 10 zusammengestellt.

Parametername	req	ind
Argument	**M**	**M**
Source Address		M
StreamID	M(=)	M(=)
Length	M(=)	M(=)
Data	M(=)	M(=)

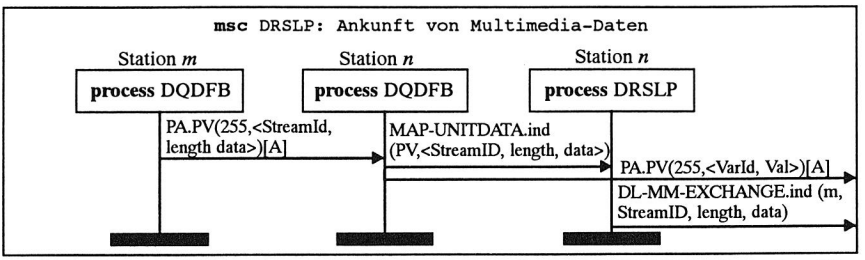
Tabelle 10: Parameter der Dienstelemente des Dienstes `DL-MM-EXCHANGE`

Die im Broadcast zu übertragen Daten sind im Parameter `Data` hinterlegt. Ihre Länge kann variieren, darf aber 244 Byte nicht überschreiten. Die Anzahl der Bytes in den aktuell übergebenen Nutzdaten ist im Parameter „Length" hinterlegt. Eine Referenzierung des Multimedia-Datenstromes, zu dem die Daten gehören, ist über den Wert des Parameters „StreamID" und die Stationsadresse des Senders („Source Address") möglich. Das Dienstelement der Anforderung dient zum Senden einer Dateneinheit. Bei einer Station eintreffende Daten, die zu Multimedia-Datenströmen gehören, werden von DRSLP unter Nutzung des `DL-MM-EXCHANGE.ind` Dienstelementes, wie in Bild 63 dargestellt, direkt über den DL_MM_SAP an den Dienstbenutzer weitergegeben.

Die Zwischenspeicherung in der Quellstation dient der Überbrückung des durch die Asynchronität des Erzeugerprozesses und des periodischen Datenaustausches möglichen zeitlichen Versatzes zwischen der Erzeugung einer neuen Dateneinheit und deren Übertragung.

Austausch von Nachrichten

Die Behandlung von Nachrichten unterscheidet sich grundsätzlich von der von Variablen. Zum einen ist ihr Inhalt beliebig, wohingegen er bei Variablen an deren Definition gebunden ist, und zum anderen werden empfangene Nachrichten nicht durch neu eintreffende bzw. neu produzierte Nachrichten überschrieben, wie dies bei Variablen der Fall ist, sondern in Sende- und Empfangseinheit jeweils in FIFO-Warteschlangen zwischengespeichert. Die zum Nachrichtenaustausch gehörigen Dienste - der Dienst zum Senden und ein lokaler Dienst zur Mitteilung über aufgetretene Fehler - werden an den diesbezüglich vorgesehenen Dienstzugangspunkten bereitgestellt. Die Angabe der Kennung eines Dienstzugangspunktes wirkt als Adresserweiterung und unterstützt den Dienstbenutzer bei der Verwaltung mehrerer logischer Verbindungen.



DL-MM-EXCHANGE.ind(m, StreamID, length, data): Dienstankündigung: Daten der Länge „length" für Datenstrom mit Kennung „StreamID" von Station „m" eingetroffen.

Bild 63: Ankunft von Dateneinheiten aus Multimedia-Strömen

Alle Anforderungen zur Übertragung von Nachrichten werden mittels eines Dienstes, durch Belegen der Parameter mit entsprechenden Werten, abgewickelt. Die Parameter dieses Dienstes (`DL-SEND-MSG`) sind in Tabelle 11 zusammenfassend dargestellt. Eine Darstellung der zugehörigen Protokolldateneinheiten ist in Anhang B.1.2 zu finden.

Parametername	req	ind
Argument	M	M
Source Address		M
Source-SAP		M
Destination Address	M	
Destination-SAP	M	
Length	M	M(=)
Ack-Mode	M	
TransferType	M	
DeadlineTime	M	
Priority	C	C(=)
Data	M	M(=)
Request-Identifier	M	

Tabelle 11: Parameter der Dienstelemente des Dienstes `DL-SEND-MSG`

Abhängig vom Wert des Dienstgüteparameters „Ack-Mode" wird eine gesicherte oder eine ungesicherte Datenübertragung mit der durch den Wert des Parameters „Priority" festgelegten Priorität durchgeführt. Handelt es sich bei der angeforderten Übertragungsart um einen gesicherten Nachrichtentransfer, so ist nur eine einzelne Station als Empfänger zulässig; anderenfalls kann für den Parameter „Destination Address" ein beliebiger Wert angegeben werden. Wird die reservierte Adresse für den Broadcast verwendet, so muß der Parameter Destination-SAP auf den Wert „15" gesetzt sein. Ansonsten kann er mit einem beliebigen Wert $\in$ [1,..,14] belegt sein. Die Länge der Nutzdaten („Data") wird durch den Wert des Parameters „Length" angezeigt. Durch entsprechendes Setzen des Parameters „TransferType" kann der Dienstbenutzer zwischen der Nutzung der periodischen und der aperiodischen Übertragungsmechanismen für Nachrichten wählen.

Im Falle der Wahl des periodischen Nachrichtenaustausches wirkt sich die Auswahl einer Prioritätsstufe nicht auf die Reihenfolge der Dienstausführung in der Sendestation aus. Es kann aber mittels des Parameters „DeadlineTime" der späteste Zeitpunkt für das Senden festgelegt werden. Für periodische Nachrichten existiert eine Beschränkung für die maximale Längen der Nutzdaten auf 27 Byte, damit die Nachricht in einem einzigen PA-Rahmen übertragen werden kann. Aperiodisch

zu übertragende Nachrichten werden in der Senderstation von der dortigen DRSLP-Instanz in je einer Warteschlange für hoch- und niederpriore Nachrichten bis zu ihrer Übertragung zwischengespeichert. Die Abarbeitung der vorliegenden niederprioren Aufträge beginnt erst dann, wenn keine hochprioren Aufträge mehr vorliegen.

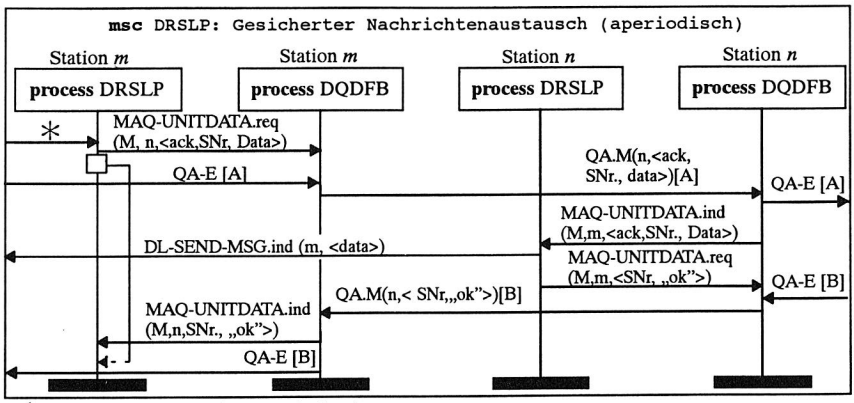
Die Abwicklung der verschiedenen Transfermodi stellt sich wie folgt dar: Bei der aperiodischen Übertragung unbestätigter Nachrichten wird eine `MAQ-UNITDATA`-Dienstdateneinheit zusammengesetzt und am QA-SAP mittels des zugehörigen Dienstes von DQDFB abgesetzt. Eine Kontrolle des Sendevorganges findet nicht statt. Der gesicherte Nachrichtenaustausch wird zwischen den zwei beteiligten Stationen unter Verwendung einer aktiven Fehlerkontrolle und Berücksichtigung von *Timeouts* [135] abgewickelt. Dazu wird von der sendenden DRSLP-Instanz eine modulo 2^6 inkrementierte Sequenznummer in den Übertragungsrahmen eingefügt. Die empfangende DRSLP-Instanz überprüft die Korrektheit der Übertragung und sendet entsprechend dem dabei erzielten Ergebnis eine positive oder eine negative Quittung an den Sender. Dabei wird die Prioritätsstufe der empfangenen Nachricht verwendet. In dieser Quittung ist die Sequenznummer der bezogenen Nachricht enthalten. Um sich gegen den vollständigen Verlust einer Nachricht zu schützen, benutzt die sendende DRSLP-Instanz zudem ein Timeout, nach dessen Ablauf eine Wiederholung des Sendens ebenso durchgeführt wird, wie nach dem Eintreffen einer negativen Quittung. Die maximale Anzahl der Wiederholungsversuche ist in der Systemkonfiguration festgelegt. Konnte eine gesichert zu übertragene Nachricht nicht korrekt an den die Partnerinstanz von DRSLP übertragen werden, so wird die lokale Dienstankündigung `DL-MSG-ERROR.ind` unter Angabe der Aufrufkennung („Request-Identifier") am Dienstzugangspunkt abgesetzt. Diese enthält als weiteren Parameter eine Spezifikation des aufgetretenen Fehlers. Der Ablauf des gesicherten Nachrichtenaustausches ist in Bild 64 dargestellt. Dabei wird von der Verwendung aperiodischer Dienste, der korrekten Übertragung der Dateneinheiten und dem rechtzeitigen Eintreffen einer Quittung ausgegangen.

Für den aperiodischen Austausch von Nachrichten ist keine zeitliche Obergrenze für die Ausführung des Sendens angebbar. Die Echtzeitkommunikationsanforderungen der höheren Schichten verlangen jedoch teilweise auch für den Nachrichtenaustausch, z.B. beim Versenden von Alarmmeldungen, nach einem zeitlichen Determinismus. Zur Erfüllung dieser Anforderung bietet DRSLP die Möglichkeit, Nachrichten einer maximalen Länge von 27 Byte in für die Station reservierten, periodisch von der BKS gesendeten, Nachrichtenrahmen zu übertragen.

DRSLP garantiert, daß wenn eine Dienstanforderung für eine im periodischen Modus zu übertragende Nachricht entgegengenommen wurde, diese auch bis zu der spezifizierten Zielzeit gesendet wird. Dazu überprüft die DRSLP-Instanz die Ausführbarkeit anhand der aktuellen Uhrzeit, der Anzahl vorliegender, noch nicht abgearbeiteter, gleichartiger Anforderungen und der Periodizität der für die Station reservierten Nachrichten. Die Zeitspanne t_w bis eine neue Anforderung gesendet werden kann, läßt sich zu einem beliebigen Zeitpunkt t für die Station mit der Adresse n dazu wie folgt berechnen:

$$t_w(t) = (k(t) + 1) \times \frac{1}{f_n} - t_l(t) \qquad (11)$$

Dabei bezeichnet $k(t)$ die Anzahl zum Zeitpunkt t bereits vorliegender gleichartiger Anforderungen, f_n die Frequenz mit der Rahmen für die periodische Nachrichtenübertragung der Station n gesendet werden und $t_l(t)$ die seit dem letzten Eintreffen eines derartigen Rahmens bei der Station bereits vergangene Zeitspanne. Der Zähler k wird nach der Entgegennahme einer Dienstanforde-



DL-SEND-MSG.req(n, aper, ack, data):	*Dienstanforderung: Gesichertes Senden einer Nachricht („Data") an die Station m im aperiodischen Modus*
MAQ-UNITDATA.req(M,n,<ack,SNr,Data>):	*Dienstanforderung: Übertragung einer Nachricht an die Station n. Zu den Nutzdaten gehört die Sequenznr. „SNr.".*
QA.M(n,<ack,SNr.,data>)[A]:	*QA-Rahmen, Typ: Nachricht, gesendet auf Bus A an Station n.*
MAQ-UNITDATA.ind(M,m,<ack,SNr,Data>):	*Dienstankündigung: Eintreffen einer Nachricht von Station m*
DL-SEND-MSG.ind(m, data):	*Dienstankündigung: Empfang einer Nachricht von Station m.*
MAQ-UNITDATA.req(M,n,<ack,SNr,Data>):	*Dienstanforderung: Senden der postivien Bestätigung an Station m. Zu den Nutzdaten gehört die Sequenznr. „SNr.".*
QA.M(n,<SNr.,„ ok">)[B]:	*QA-Rahmen, Typ: Nachricht, gesendet auf Bus B an Station m.*
MAQ-UNITDATA.ind(M,m,SNr,„ ok">):	*Dienstankündigung: Eintreffen einer Nachricht von Station m*

Bild 64: Gesicherter, aperiodischer Austausch von Nachrichten

rung zum Austausch einer Nachricht im periodischen Modus von DRSLP inkrementiert und nach der Entgegennahme der zugehörigen `MA-SENT.ind` Dienstankündigung dekrementiert. Kann die geforderte Zielzeit nicht eingehalten werden, so wird die Dienstanforderung mit einer entsprechenden Fehlermeldung zurückgewiesen.

Unabhängig vom Transfermodus werden bei einer Station eingetroffene, korrekt übertragene Nachrichten mittels der Dienstankündigung `DL-SEND-MSG.ind` an dem im Parameter „Destination-SAP" spezifizierten Dienstzugangspunkt an den Dienstbenutzer übergeben.

5.5 Zusammenfassung der wesentlichen Eigenschaften von DQDFB und DRSLP

DQDFB stellt mit den Diensten und Protokollmechanismen eine Basis für den periodischen und aperiodischen Austausch von Daten zwischen beliebigen Stationen dar. An der Diensteschnittstelle kann von der verwendeten Topologie bereits abstrahiert werden. Die Aufnahme eines Mechanismus zur Segmentierung und Reassemblierung von längeren Nachrichten schafft auch eine Transparenz bezüglich der festen Rahmenlänge, ohne den benötigten zeitlichen Determinismus des periodischen Datenaustausches zu beeinträchtigen. Die aktive Anschaltung aller Stationen erlaubt eine Freigabe nicht mehr benötigter, belegter Übertragungsrahmen, wodurch die Bandbreitenausnutzung insgesamt verbessert wird. Die Adressierung einzelner Stationen und von Gruppen von Stationen wird unterstützt. Die Einführung von speziellen Rahmentypen für den Austausch von Nachrichten und Variablen macht eine kurze Latenzzeit der aktiv an das Medium angeschlossenen Stationen möglich und bildet die Basis für die Kommunikation der als Dienstbenutzer von DQDFB auftretenden Instanzen von DRSLP.

DRSLP beinhaltet Funktionen zum periodischen und aperiodischen Austausch von Daten und bietet alle in Anforderung 8 (vgl. Kapitel 3.1) geforderten PDU-Sequenzen. Je nach Semantik der Daten werden unterschiedliche Dienste für deren Austausch zur Verfügung gestellt, wobei sowohl für Prozeßvariable und Dateneinheiten aus Multimedia-Datenströmen als auch für Nachrichten die Berücksichtigung von Echtzeit-Anforderungen bei der Kommunikation erfolgt. Für Prozeßvariable wird durch die verteilten Instanzen von DRSLP zudem die verteilte Datenbasis verwaltet.

Das Zusammenspiel der Protokollinstanzen von DQDFB und DRSLP wird in den Tabellen B.2 bis B.4 in Anhang B dargestellt. Die Mechanismen der Sicherungsschicht verbunden mit einer adäquaten Ausprägung der Bitübertragungsschicht stellen damit eine funktional auf die Belange der prozeßnahen Kommunikation ausgerichtete Basis für Protokolle höherer Schichten dar. Es ist aber auch möglich, Applikationen direkt auf die Schnittstelle von DRSLP aufzusetzen.

6 Protokolle der OSIRIS-Anwendungsschicht

In OSIRIS sind für die zwischen der Sicherungs- und der Anwendungsschicht liegenden Schichten des ISO-OSI Basisreferenzmodells keine Protokollausprägungen notwendig. Nachfolgend werden die Protokolle der Anwendungsschicht beschrieben, die somit direkt auf das Protokoll DRSLP der Sicherungsschicht aufsetzen.

6.1 Schnittstelle zwischen Anwendungs- und Sicherungsschicht

Von dem Protokoll der Anwendungsschicht (vgl. Kap. 6.2) wird die Bereitstellung von Mechanismen zur verbindungslosen und -orientierten Nachrichtenkommunikation sowie eine Zugriffsschnittstelle zur verteilten Datenbasis benötigt. Diese Funktionalität wird durch die Protokolle der Sicherungsschicht nur teilweise erbracht. Insbesondere fehlt dort noch eine Unterstützung für die direkte Kommunikation zwischen Instanzen des Protokolls der Anwendungsschicht über logische Kommunikationsbeziehungen. Diese Funktionen werden in OSIRIS durch das Protokoll LLI bereitgestellt. Die Namensgebung und die Bezeichnung der Dienste orientieren sich an den Definitionen des entsprechenden Protokolls in PROFIBUS. Die Art der Diensterbringung und die Verbindungsverwaltung unterscheiden sich jedoch beträchtlich, wobei in OSIRIS im wesentlichen eine Vereinfachung des Systems bei gleichzeitiger Erhöhung der Flexiblität angestrebt wurde. Zur Erbringung der angebotenen Funktionalität nutzt jede LLI-Instanz die Dienste der Sicherungsschicht und kommuniziert mit ihrer LLI-Partnerinstanz.

6.1.1 Modellvorstellung für das LLI-Protokoll

Für den Austausch von Informationen zwischen Stationen werden Nachrichten und die verteilte Datenbasis verwendet. Nachrichten werden zwischen den Applikationen über Kommunikationsbeziehungen ausgetauscht, die eindeutig zugeordnete Kommunikationsendpunkte haben und verbindungsorientiert oder verbindungslos sein können. Bei der verbindungsorientierten Kommunikation werden die *Verbindungsaufbauphase*, die *Datentransferphase* und die *Verbindungsabbauphase* unterschieden. Diese Art der Kommunikation findet in OSIRIS grundsätzlich nur zwischen genau zwei Applikationen statt, wobei die Übertragung von Nutzdaten zwischen den Kommunikationspartnern nur in der Datentransferphase möglich ist.

Bei der verbindungslosen Kommunikation ist der Nachrichtenaustausch auch im Multicast möglich, wobei dann grundsätzlich nur die Nutzung unbestätigter Kommunikationsdienste erlaubt ist. Nachrichten, die im Multicast gesendet wurden, werden in den Empfängerstationen an die in der Station aktuell laufenden Applikationen weitergeleitet. Dabei kann eine Applikation mehrere Kommunikationsbeziehungen gleichzeitig verwalten. Verbindungsorientierte Kommunikationsbeziehungen müssen zur Laufzeit des Systems explizit aufgebaut werden; verbindungslose Kommunikationsbeziehungen hingegen befinden sich grundsätzlich in der Datentransferphase. Für den Zugriff auf die verteilte Datenbasis wird keine Kommunikationsbeziehung benötigt.

Eine beispielhafte Konfiguration ist in Bild 64 dargestellt. In dieser Konfiguration bestehen verbindungsorientierte Kommunikationsbeziehungen zwischen den Stationen „0" und „1", „1" und „3" sowie „2" und „3". Die Station „1" kann über ihren DL_MSG_SAP „11" Nachrichten an alle Stationen und über einen anderen DL_MSG_SAP („5") Nachrichten im Multicast an eine Gruppe von Stationen, bestehend aus den Stationen mit den Kennungen „2" und „3", senden. In beiden Fällen handelt es sich um verbindungslose Kommunikation. Alle Applikationen haben Zugriff auf die verteilte Datenbasis und können über diese Informationen mittels der von den Protokollen der Sicherungsschicht angebotenen Dienste austauschen.

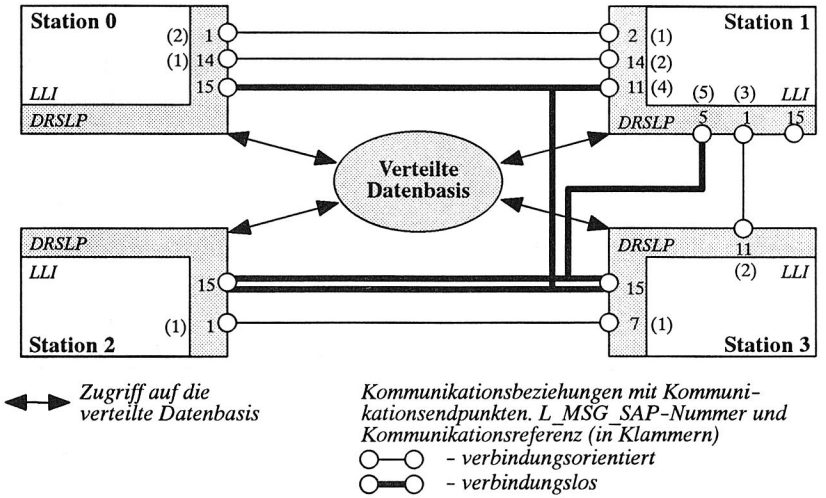


Bild 64: Kommunikation zwischen Stationen kann direkt über Applikationsbeziehungen oder indirekt über die verteilte Datenbasis erfolgen

Das LLI stellt seine Dienste an drei Dienstzugangspunkten zur Verfügung: Dem „LLI-SAP 0" für den Austausch von Nachrichten über Kommunikationsbeziehungen, dem „LLI-SAP 1" zum Zugriff auf die gemeinsame Datenbasis und dem „LLI-SAP 2" zur Abwicklung der für das Management benötigten Kommunikation. Es verwaltet die Kommunikationsbeziehung für das Management, die über den DL_MSG_SAP mit der Kennung „0" abgewickelt wird, sowie maximal 14 zusätzliche verbindungsorientierte Kommunikationsbeziehungen. Alle Kommunikationsbeziehungen werden von den Applikationen über *Kommunikationsreferenzen* identifiziert. Jede Kommunikationsreferenz ist genau einem DL_MSG_SAP der Sicherungsschicht zugeordnet. Einem DL_MSG_SAP dürfen dabei mehrere Kommunikationsreferenzen zugeordnet sein, darunter jedoch nur eine für verbindungsorientierte Kommunikation. Für die Kommunikationsbeziehung zum Austausch von Managementinformationen ist die Kommunikationsreferenz „0" reserviert. Die anderen Kommunikationsbeziehung sind nicht vorprojektiert und können zur Laufzeit des Systems auf- und abgebaut werden. Für aufgebaute Kommunikationsbeziehungen besteht ein Eintrag in der als Verwaltungsstruktur dienenden *Kommunikationsbeziehungsliste*. Dieser Eintrag besteht u.a. aus der Stationsadresse und der Kennung des DL_MSG_SAP des Kommunikationspartners sowie der Kennung des lokalen DL_MSG_SAPs. Über diese Adressinformation kann bei verbindungsorientierten Kommunikationsbeziehungen eine Anwendungsinstanz eindeutig referenziert werden. Bezugnehmend auf die in Bild 64 dargestellte Beispielkonfiguration hätte die Station „1" die in Bild 65 dargestellten Einträge in der Kommunikationsbeziehungsliste.

Sollen Nachrichten von einer Station im Multicast gesendet werden, so bestehen dafür ebenfalls Einträge in der Kommunikationsbeziehungsliste. Als Zieladresse kann eine beliebige Adresse angegeben sein. Die Nachrichten werden in der bzw. in den Zielstation(en) grundsätzlich an den für die Multicast-Kommunikation reservierten DL_MSG_SAP mit der Kennung „15" gesendet. In

dem in Bild 64 dargestellten Beispiel existieren Kommunikationsbeziehung für den Nachrichtenaustausch an eine Gruppe von Stationen (Gruppenadresse „191") bzw. an alle Stationen (Broadcastadresse „255") mit den Kommunikationsreferenzen „4" bzw. „5". Seitens der sendenden Station kann ein beliebiger DL_MSG_SAP mit Ausnahme dessen, der für den Austausch von Managementinformationen reservierten ist, verwendet werden.

Kommunikations-referenz	*Adresse der Zielstation*	*Entfernter DL_MSG_SAP*	*Lokaler DL_MSG_SAP*
0	128	0	0
1	0	1	2
2	0	14	14
3	3	11	1
4	191	15	11
5	255	15	5

Bild 65: Beispielhafter Ausschnitt aus einer Kommunikationsbeziehungsliste

Das RTSIMS-Protokoll, auf das in Kapitel 6.2 näher eingegangen wird, basiert auf dem *Client-Server* Modell [38]. In diesem Protokoll ist die Initiative zum Verbindungsaufbau dem Client zugeordnet. Die Zuordnung von DL_MSG_SAPs des Servers zu der bzw. den dort laufenden Applikation(en) wird während der Konfigurierung des Servers vorgenommen. Die Information, über welchen DL_MSG_SAP die gewünschte Serverapplikation einer Station ansprechbar ist, liegt damit bei der Erstellung der Client-Applikation vor. Weiterführende Adressierungsmechanismen, wie sie aus der OSI-Umgebung für Anwendungs-Arbeitseinheiten [25, 137] (Application-Entities, AE) und deren Identifikatoren (AE-qualifiers) bekannt sind, werden aufgrund der hier sinnvollen statischen Zuordnung von Servern zur realen Geräten und der Flexibilität des RTSIMS-Protokolls nicht benötigt.

6.1.2 Verwendetes Protokoll und Diensteschnittstelle

Die vom LLI angebotenen Dienste lassen sich nach ihrem Verwendungszweck klassifizieren: Es werden dabei unterschieden:

- Lokale Dienste,
- Dienste zur Verwaltung von Verbindungen,
- Dienste zum Austausch von Nachrichten während der Datentransferphase,
- Dienste zum Zugriff auf die verteilte Datenbasis und
- Dienste zum Austausch von Dateneinheiten aus Multimedia-Datenströme.

Lokale Dienste

Im Client und im Server müssen Kommunikationsreferenzen für die verbindungslose und -orientierte bzw. nur für die verbindungslose Kommunikationsbeziehungen dynamisch erzeugt und wieder freigegeben werden können. Für diesen Zweck sind dort die lokalen Dienste

- `LLI-GET-CR/LLI-RELEASE-CR`

definiert. Die Parameter des lokalen Dienstes `LLI-GET-CR` (CR: Communication Reference) sind in Tabelle 12 dargestellt.

Parameter Name	req	cnf
Argument	**M**	
Remote Address	M	
Remote DL_MSG_SAP	C	
Result(+)		**S**
CR-ID		M
Result(-)		**S**
Error Specification		M

Tabelle 12: Parameter der Dienstelemente des Dienstes LLI-GET-CR

In der Anforderung wird die Adresse des gewünschten Kommunikationspartners im Parameter „Remote Address" angegeben. Handelt es sich dabei um die Adresse einer einzelnen Station (Adresse < 128), so ist zudem die Angabe des dort anzusprechenden Dienstzugangspunktes mittels des Parameters „Remote DL_MSG_SAP" notwendig. Die lokale LLI-Instanz wählt einen freien DL_MSG_SAP aus, reserviert ihn für diese Kommunikationsbeziehung und übergibt die Identifikation der Kommunikationsreferenz („CR-ID") in der lokalen Dienstbestätigung an die anfordernde Applikation. Sind bereits alle DL_MSG_SAPs reserviert, so wird eine negative Quittung an den Requester übergeben. Wird eine Kommunikationsreferenz für eine verbindungslose Kommunikationsbeziehung angefordert, so wird von der LLI-Instanz ein beliebiger DL_MSG_SAP zugeordnet, der auch für weitere Kommunikationsbeziehungen genutzt werden kann. Positiv bestätigte Anforderungen werden in die Kommunikationsbeziehungsliste aufgenommen.

Mittels des Dienstes `LLI-RELEASE-CR` können die für eine Kommunikationsbeziehung reservierten Ressourcen unter Angabe der Kommunikationsreferenz freigegeben werden. Eine negative Quittierung erfolgt dann, wenn es sich um eine verbindungsorientierte Kommunikationsbeziehung handelt, die sich in der Datentransferphase befindet.

Zur Abwicklung von Dienstanforderungen mit Zielzeit ist für den Fall, daß von der DRSLP-Instanz die `MAP-UNITDATA`-Dienstanforderung des LLI abgewiesen wird, der lokale Benachrichtigungsdienst `LLI-MSG-ERROR` definiert. Er beinhaltet neben einer Spezifikation des aufgetretenen Fehlers die Aufrufkennung der verursachenden `LLI-DTU`-Dienstanforderung (s.u.).

Dienste zur Verwaltung der Verbindung

Zum Auf- und Abbau verbindungsorientierter Kommunikationsbeziehungen sind folgende Dienste definiert:

- `LLI-ASSOCIATE`: Aufbau einer verbindungsorientierten Kommunikationsbeziehung. Dieser bestätigte Dienst wird nur vom Client aufgerufen und beinhaltet die in Tabelle 13 aufgeführten Parameter.
- `LLI-ABORT`: Abbau einer verbindungsorientierten Kommunikationsbeziehung. Dieser Dienst ist unbestätigt und kann von Client oder Server aufgerufen werden. Nach dem Senden baut der Requester, nach dem Empfang der Responder die Kommunikationsbeziehung ungeachtet evtl. noch ausstehender Dienstbestätigungen lokal ab.

Beim Dienst `LLI-ASSOCIATE` werden an der Diensteschnittstelle grundsätzlich die Kommunikationsreferenz und die Nutzdaten, die durch Protokolldateneinheiten des Dienstes `INITIATE` des RTSIMS-Protokolls gebildet werden, übergeben. Bei den Dienstelementen der Anforderung und der Ankündigung wird zusätzlich die Spezifikation des Transfermodus (periodisch oder aperiodisch) übergeben. Im Falle einer angeforderten aperiodischen Datenübertragung ist zudem das

Setzen des Parameters für die gewünschten Priorität der Dienstausführung (hoch bzw. niedrig) notwendig.

Dienste zum Austausch von Nachrichten

Für den Austausch von Nachrichten während der Datentransferphase und im Rahmen des Systemmanagements sind drei Dienste definiert:

- `LLI-DTC` (Data Transfer Confirmed): Dieser Dienst erlaubt den Austausch von Daten bestätigter Anwendungsdienste über eine verbindungsorientierte Kommunikationsbeziehung.
- `LLI-DTA` (Data Transfer Acknowledged): Mit Hilfe dieses Dienstes können unbestätigte Anwendungsdienste auf einer verbindungsorientierten Kommunikation genutzt werden. Dabei wird die Übertragung der Daten überwacht.
- `LLI-DTU` (Data Transfer Unconfirmed): Dieser Dienst ist für die ungesicherte Übertragung von Daten für unbestätigte Dienste auf verbindungsorientierten oder -losen Kommunikationsbeziehungen vorgesehen und erlaubt die Angabe einer Zielzeit für das Senden der Dateneinheit.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Communication Reference	M	M(=)		
Data	M	M(=)		
Transfer Mode	M	M(=)		
Priority	C	C(=)		
Result(+)			**S**	**S**
Communication Reference			M	M(=)
Data			M	M(=)
Transfer Mode			M	
Priority			C	
Result(-)			**S**	**S**
Communication Reference			M	M(=)
Data			M	M(=)
Transfer Mode			M	
Priority			C	

Tabelle 13: Parameter der Dienstelemente des Dienstes `LLI-ASSOCIATE`

Die Parameter der Dienstelemente dieser drei Dienste entsprechen weitgehend denen des Dienstes `LLI-ASSOCIATE`, wobei für die Dienste `LLI-DTA` und `LLI-DTU` nur die Dienstelemente der Anforderung und der Ankündigung definiert sind. Für den Dienst `LLI-DTU` ist zusätzlich beim Senden die Angabe einer Zielzeit („Target Time") optional möglich. Ergänzend dazu steht für das Attribut „Priority" zusätzlich der Wert „real time" zur Verfügung. Diese Prioritätsstufe wird zur Anzeige des Wunsches, den Dienst bis zu einer bestimmten Zielzeit auszuführen, gesetzt. In diesem Ausführungsmodus muß eine eindeutige Aufrufkennung („Request Identifier") übergeben werden.

Wie auch die Dienste `LLI-ASSOCIATE` und der `LLI-ABORT` werden die Dienste `LLI-DTC`, `LLI-DTA` und `LLI-DTU` auf den Dienst `DL-SEND-MSG` von DRSLP abgebildet. Nach Absetzen einer Anforderung bzw. der Antwort werden aus der Kommunikationsbeziehungsliste die für die spezifizierte Kommunikationsreferenz benötigten Adressinformationen entnommen. Sie dienen zusammen mit den Werten der Aufrufparameter des LLI-Dienstes zum Setzen der Parameter der DRSLP-Dienstanforderung. Der Ablauf eines LLI-DTC-Dienstes ist in Bild 69 im Zusammenhang mit der Ausführung eines bestätigten Dienstes von RTSIMS dargestellt.

Dienste zum Zugriff auf die verteilte Datenbasis und Multimedia-Datenströme

Die vom DRSLP angebotenen Dienste zum Zugriff auf Variablen werden durch das LLI-Protokoll hindurchgereicht und stehen dort mit den gleichen Parametern unter den Bezeichnungen `LLI-WRITE-LOC`, `LLI-READ-LOC`, `LLI-UPDATE` und `LLI-PUBLISH` am LLI-SAP mit der Kennung „1" zur Verfügung. Zum Schreiben von Dateneinheiten, die zu Multimedia-Datenströmen gehören, wird der Dienst `LLI-MM-PUT` als direkte Abbildung des entsprechenden Dienstes von DRSLP angeboten.

Von DRSLP werden am DL_VAR_SAP nach dem Senden und dem Empfangen von Variablen und am DL_MM_SAP nach dem Empfang einer zu einem Multimedia-Datenstrom gehörigen SDU Dienstankündigungen erzeugt. Im Regelfall muß nur eine Teilmenge dieser Dienstankündigungen vom LLI an die Applikationen zugestellt werden. Dabei kann es aber vorkommen, daß bestimmte Dienstankündigungen an mehrere Applikationen weitergegeben werden müssen. Deshalb ist im Stationsmanagement ein Mechanismus vorgesehen, der das Sperren und die Freigabe von Dienstankündigungen für einzelne Applikationen ermöglicht (vgl. Kap. 7.5.2). Freigegebene Dienstankündigungen werden mittels der Dienstankündigungen `LLI-SENT`, `LLI-RECEIVED` und `LLI-MM-RECEIVED` an die Applikationen weitergegeben. Die Parameter dieser Dienstankündigungen entsprechen denen der Dienstankündigungen von DRSLP. Diese Selektion verhindert eine unerwünschte Belastung der Protokollinstanz der Anwendungsschicht durch für die Anwendung nicht relevante Dienstankündigungen.

Für eine Implementierung sind die Dienste zur Verbindungsverwaltung, zum Zugriff auf die verteilte Datenbank sowie die Dienste `LLI-DTC` und `LLI-DTU` zwingend vorgeschrieben.

6.1.3 Zusammenfassung der wesentlichen Eigenschaften

Das LLI stellt damit zusammenfassend betrachtet Mechanismen zu einer einfachen und flexiblen Verbindungsverwaltung, dem Zugriff auf die verteilte Datenbasis und den Austausch von zu Datenströmen gehörenden Dateneinheiten bereit, die von den Protokollen der Anwendungsschicht genutzt werden können, und schafft für diese eine weitgehende Transparenz bezüglich der verwendeten Kommunikationsmechanismen.

6.2 Benutzerschnittstelle der Anwendungsschicht

Anwendungen tauschen untereinander Daten mit Hilfe des Protokolls und der Dienste der Anwendungsschicht aus. Zur Erfüllung der analysierten Anforderungen müssen die Dienste und das zugehörige Protokoll insbesondere in der Lage sein, dem Dienstbenutzer Mechanismen zur

- Behandlung von zeitgestempelten Daten,
- Echtzeitkommunikation zwischen Anwendungen,
- Unterstützung einer Autonomie der einzelnen Stationen und zur
- Steuerung von abstrakt modellierten Multimedia-Aufnahme- und Feldgeräten sowie der zugehörigen Datenströme anzubieten.

Wie in den Bewertungen der betrachteten verfügbaren Systeme ausgeführt (vgl. Kap. 3.4 bzw. 4), erfüllen die für die Systeme PROFIBUS, FIP und OLCHFA definierten Protokolle der Anwendungsschicht diese Anforderungen in einem nur unzureichenden Maße. Das Protokoll RTSIMS der Anwendungsschicht von OSIRIS kann diese Anforderungen auf Basis der Mechanismen der darunterliegenden Protokolle und der eigenen Funktionalität entsprechen. Dieses Protokoll und die zugehörigen Dienste werden im folgenden näher beschrieben. Vorangestellt wird dabei die Erläuterung der wesentlichen Grundprinzipien.

6.2.1 Grundprinzipien: Client-Server Modell und Objektorientierung

Der nachrichtenorientierte Bestandteil von RTSIMS basiert – wie auch das FMS-Protokoll aus PROFIBUS und das MMS-Protokoll – auf dem *Client-Server*-Modell, das die hierarchische Gliederung der Steuerungsstruktur und der Verteilung der Aufgaben in der rechnerintegrierten Fertigung widerspiegelt. Der *Server* stellt das System dar, das auf Aufforderung durch den *Client* Aktionen durchführt und gegebenenfalls Ergebnisse meldet. Ein Server kann beispielsweise durch ein Feldgerät mit der zugehörigen Gerätesteuerung gebildet werden, wohingegen Clients durch Applikationen hierarchisch höher angesiedelter Steuerungssysteme, z.B. der Zellensteuerung, repräsentiert werden. Im Sinne des ISO-OSI Basisreferenzmodells stellen sowohl Client als auch Server *Anwendungsinstanzen* dar. Es besteht dabei durchaus die Möglichkeit, daß ein einzelnes System sowohl als Server als auch als Client auftritt. Der Ablauf der Kommunikation zwischen Client und Server ist im wesentlichen durch das Verhalten des Servers als Diensterbringer geprägt und die Beschreibung der Dienste und Objekte wird – sofern nicht explizit anders dargestellt – aus dessen Sicht vorgenommen. Bezüglich des Austausches von Daten über die verteilte Datenbasis sind alle Stationen gleichberechtigt. Es wird hier bei jeder einzelnen Variablen zwischen deren *Erzeuger* und deren *Konsumenten* unterschieden.

In FMS und MMS stellt die Verwendung von Objekten und von Funktionen, die auf diese Objekte wirken, ein weiteres wesentliches Entwurfs- und Beschreibungsprinzip dar. Nach den Paradigmen der Methodik der Objektorientierung kommt dabei jedoch nur ein *objektbasierter* Ansatz zur Anwendung [37]. RTSIMS basiert hingegen auf der *Objektorientierung*, d.h., es wurden auch die Mengenabstraktion und die Vererbung berücksichtigt. Die Verwendung dieses Ansatzes zur Beschreibung der *Kommunikationsobjekte* des Servers führt im Vergleich mit z.B. MMS bei Beschränkung auf eine gleichwertige Funktionalität zu einer deutlichen Reduzierung der Anzahl benötigter Dienste. Dies wird durch die Ausnutzung gemeinsamer Eigenschaften verschiedener Objekte, z.B. bei deren Erzeugung oder Zustandsabfrage, ermöglicht. Kommunikationsobjekte repräsentieren Anwendungsobjekte, wie z.B. einen Zähler oder einen Steuerungsprozeß. Die Abbildung der Kommunikationsobjekte auf die Anwendungsobjekte und vice versa ist implementierungsabhängig.

Die in dieser Arbeit benutzte Terminologie und die zur Veranschaulichung von Objektbeziehungen und Vererbungshierarchien gewählte graphische Darstellungsform basieren auf den Ausführungen von *Martin et al.* [129]. Diese nutzen den Begriff des *Objekttyps* zur konzeptionellen Beschreibung einer Kategorie von Objekten und den der *Operation* zur Beschreibung der Manipulation der Daten eines Objektes. Objekttypen werden in einer Implementierung zu *Objektklassen*, von denen durch Instantiierung *Objekte* erzeugt werden, auf die *Methoden* als Realisierung von Operationen wirken.

Objekte repräsentieren damit in RTSIMS das nach außen sichtbare Verhalten eines realen Systems und erlauben eine Abstraktion [138] von dessen Ausprägung. Die für einen Objekttyp definierten Operationen werden durch Dienste repräsentiert, die vom Server über das Kommunikationssystem angeboten werden. Diese Abstraktion erlaubt damit die Verwendung der gleichen Dienste zur Kommunikation mit unterschiedlichen Feldgeräten bzw. Multimedia-Aufnahmegeräten, wie dies in Bild 66, beschränkt auf die erstgenannte Klasse von Geräten, dargestellt ist.

Während der Konfigurierung des Servers werden diesem Kommunikationsreferenzen, und diesen wiederum lokale Dienstzugangspunkte der Sicherungsschicht fest zugewiesen. Damit wird eine Adressierung der Serverapplikation für Clients möglich. Die Implementierung des Servers kann

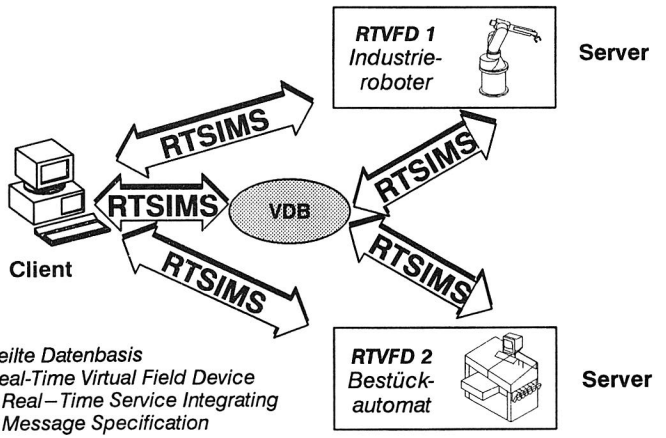


Bild 66: Abstraktion durch Verwendung von virtuellen Feldgeräten

darüber hinaus für bestimmte Kommunikationsreferenzen eine Beschränkung des Spektrums dort angebotener Dienste vorsehen. Damit wird es einerseits möglich, den Zugriff auf bestimmte Ressourcen des Servers, z.B. die Steuerung der Programmausführung, exklusiv an eine eingeschränkte Menge von Client-Applikationen zu vergeben, und andererseits kann verschiedenen Systemen der Zugriff auf gemeinsam nutzbare Ressourcen, wie z.B. Daten, gewährt werden.

Die lokale Management-Informations-Basis (MIB) einer jeden Station beinhaltet ein Objektverzeichnis, mit dessen Hilfe die Zuordnung von logischen Bezeichnern zu deren physischen Adressen nachvollzogen werden kann. Dieses Objektverzeichnis wird während der Konfigurierung des Netzwerkes angelegt und während der Laufzeit des Systems bei der Instantiierung oder Löschung von Objekten angepaßt.

Dienstmodell

Die Grundlagen bezüglich bestätigter und unbestätigter Dienste und deren Darstellung wurden bereits in Kapitel 4.1.2 eingeführt. Zusätzlich zu den bei den einzelnen Dienstelementen dargestellten Parametern sind an der Aufrufschnittstelle noch die Kommunikationsreferenz und – bei bestätigten Diensten mit Ausnahmen des Dienstes `Initiate` – die durch den Benutzer zu verwaltende Aufrufkennung („Invoke-ID") anzugeben. Diese Aufrufkennung dient der Identifikation der zu einer Dienstanforderung gehörigen Antwort. Sie wird vom Requester vorgegeben und vom Responder für die zugehörige Antwort übernommen.

Soweit nicht explizit anders erwähnt, verfügen alle Dienste an der Diensteschnittstelle zudem über die im einzelnen nicht mehr aufgeführten Parameter „Priority" und „Transfer-mode". Der Wert des Parameters „Priority" wirkt sich auf die Reihenfolge der Abarbeitung der Dienstanforderungen in den unteren Schichten aus. Der Wertebereich umfaßt die Werte „high", „low" und „real time" zur Anzeige einer gewünschten hoch- oder niederprioren Dienstausführung bzw. einer solchen, die mit einer Zielzeit versehen ist. Die Anforderung der Ausführung bis zu einem gegebenen Zielzeitpunkt ist dabei nur für unbestätigte Dienste zulässig. Über dem Wert des Parameters „Transfer-mode" kann der Dienstbenutzer steuern, ob die Anforderungen mittels der periodischen oder der aperiodischen Nachrichtenübertragung und ungesichert oder gesichert übertragen wer-

den sollen. Beide Parameter beschreiben damit die geforderte Güte einer Dienstausführung. Bestätigte Dienste werden grundsätzlich unter Verwendung des LLI-DTC-Dienstes abgewickelt. Unbestätigte Dienste werden je nach Wert der Dienstgüteparameter auf den LLI-DTA- (gesicherte Datenübertragung) oder den LLI-DTU-Dienst (sonstige Fälle, insbesondere mit Zielzeit) abgebildet. Der lokale Benachrichtungsdienst LLI-MSG-ERROR ist dem Dienst REJECT zugeordnet.

Die Dienstankündigungen der Dienste, die für den Datenaustausch über die verteilte Datenbasis vorgesehen sind, werden wie folgt abgebildet: Für die Dienstankündigungen LLI-SENT.ind und LLI-RECEIVED.ind wird eine Abbildung auf die Dienstankündigungen der Schreib- bzw. die Dienstbestätigungen der Lesedienste vorgenommen. Dienstankündigungen des Dienstes LLI-MM-RECEIVED werden als Dienstankündigung des Dienstes MM-EXCHANGE an den Dienstbenutzer weitergeleitet. Eine sinngemäße Abbildung erfolgt auch für die jeweiligen Dienstanforderungen.

6.2.2 Verwaltung logischen Applikationsverbindungen

Auf Ebene der Applikation werden logische Verbindungen zwischen Anwendungsprozessen zum Austausch von bestätigten Nachrichten benötigt. Diese werden mittels des Dienstes INITIATE auf- und mittels des Dienstes ABORT wieder abgebaut. Als Dienstparameter der Anforderung wird das Attribut „Supported Services" übergeben (vgl. Tabelle 14). Diese Bitfolge enthält Informationen darüber, welche der nicht zwingend zu unterstützenden Dienste (vgl. Kap. 6.4) durch die Implementierung des Requesters unterstützt werden.

Parameter Name	req	ind	rsp	cnf
Argument	**M**	**M(=)**		
Supported Services	M	M(=)		
Priority	M	M(=)		
Result(+)			**S**	**S(=)**
Supported Services			M	M(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 14: Parameter der Dienstelemente des Dienstes INITIATE

In einer positiven Dienstbestätigung sind Informationen enthalten, die angeben, welche der Dienste durch beide Kommunikationspartner unterstützt werden. Der Responder erzeugt diese Information durch Schnittmengenbildung aus der Menge der durch den Requester bzw. durch ihn selbst unterstützten Dienste. Eine negative Dienstbestätigung beinhaltet eine Spezifikation des Fehlers. Dieser Dienst wird von Servern nur als Responder unterstützt.

Der unbestätigte Dienst ABORT kann sowohl vom Client als auch vom Server aufgerufen werden und führt zu einem sofortigen Abbau der Kommunikationsbeziehung. Als einziger Parameter wird die optionale Angabe einer Fehlerspezifikation in der Anforderung und der Ankündigung geführt.

Der Dienst Reject dient der Rückweisung einer nicht korrekten Protokolldateneinheit und enthält eine Spezifikation des festgestellten Fehlers sowie die Aufrufkennung des Dienstes, der den Fehler verursacht hat. Neben der Meldung von ungültigen Protokolldateneinheiten aufgrund syntaktischer Fehler, z.B. zu lange Nutzdaten, oder semantischer Fehler, z.B. Aufruf eines nicht unterstützten Dienstes, wird dieser Dienst auch dann genutzt, wenn für einen anderen Dienst die für dessen Ausführung vorgegebene Zielzeit nicht eingehalten werden kann.

6.2.3 Objekte des echtzeitfähigen und dienstintegrierenden Protokolls RTSIMS

Zur Beschreibung eines, u.U. mit einem Multimedia-Aufnahmegerät ausgestatteten, weitgehend autonom agierenden, realen Feldgerätes werden Objekte (vgl. Bild 67) und Dienste benötigt, die es erlauben,

- das Feldgerät mit seinen Attributen an sich,
- die dort zur Steuerung des Systems eingesetzten Programme und deren Verwaltungsmechanismen,
- die (Echtzeit-) Prozeßvariablen,
- Ereignisse im Fertigungsprozeß sowie
- die Multimedia-Aufnahmegeräte zu modellieren.

Zur Modellierung des Feldgerätes an sich dient die Objektklasse des *echtzeitfähigen, virtuellen Feldgerätes* (*RTVFD*). Pro Server existiert genau eine statische Instanz dieser Objektklasse, d.h., es kann weder eine neue Instanz gebildet, noch die vorhandene gelöscht werden. Programme, mit deren Hilfe das reale Gerät gesteuert wird, z.B. das Programm zur Steuerung des Bestückkopfes eines Bestückautomaten, werden durch Programminstanzen (*Program Invocations, PI*) modelliert und bestehen aus einem oder mehreren Datenblöcken (*Domains*), die beispielsweise das Maschinenprogramm, Werkzeugdaten und Korrekturwerte enthalten. Die Instanzen von Domains stellen selbst wiederum Kommunikationsobjekte dar und sind ebenso wie die PIs dynamischer Natur, d.h., sie können zur Laufzeit des Systems erzeugt und wieder gelöscht werden. Der alarm- oder zeitgesteuerte Aufruf von PIs wird entsprechend den Bedürfnissen der Applikation durch die statische Instanz des *Scheduler*-Objekts eines Servers vorgenommen. Die Informationen, wie der Ablauf zu organisieren ist, werden für den Scheduler in Tabellen gehalten (*Scheduling Tables*), die als Objektinstanzen dynamischer Natur sind und aus mindestens einem Domain bestehen.

Prozeßvariable und Echtzeit-Prozeßvariable (*Variable* bzw. *RT-Variable*) gehören wiederum zu den Objektklassen mit ausschließlich statischen Instanzen. Sie stellen mit den für sie definierten Diensten die Schnittstelle der Anwendung zur verteilten Datenbasis dar.

Ausnahme- und Fehlerzustände des technischen Prozesses ziehen in der Regel eine Veränderung des Wertes von Prozeßvariablen nach sich. Die Erkennung von Ereignissen dem Kommunikationsprotokoll der Anwendungsschicht zu überlassen, ist deshalb insbesondere dann naheliegend, wenn die Werte von Prozeßvariablen – wie in OSIRIS der Fall – über eine verteilte Datenbasis ausgetauscht werden. Dieses bereits für das OLCHFA-System in ähnlicher Form realisierte Konzept wird für RTSIMS übernommen. Zu diesem Zweck wird der Objekttyp des Ereignisbedingung (*Event Condition*) eingeführt, dessen grundsätzlich statische Instanzen alle notwendigen Informationen zur Beschreibung einer Ereignisbedingung und Parameter für die nach der Erkennung eines Ereignisses anzustoßende Aktion beinhalten.

Dem zunehmenden Einsatz visueller und auditiver Sensorik und der Forderung nach der Übertragung der von diesen Systemen abgreifbaren multimedialen Information Rechnung tragend, wird das Konzept des virtuellen Multimedia-Gerätes (Virtual MultiMedia Device, VMMD) eingeführt. Das VMMD modelliert die gemeinsamen Eigenschaften verschiedenartiger realer Systeme, wie z.B. einer Kamera und eines Mikrofons, deren Spezifika in detaillierteren Untertypen des VMMD Berücksichtigung finden. Wie das RTVFD, so sind auch die VMMDs statische Objekte des Servers, die auf Anforderung von Clients deren Aufträge ausführen. Da derartige Informationen häufig von mehreren Clients gleichzeitig benötigt werden, sind im Server Vorkehrungen zur Verwal-

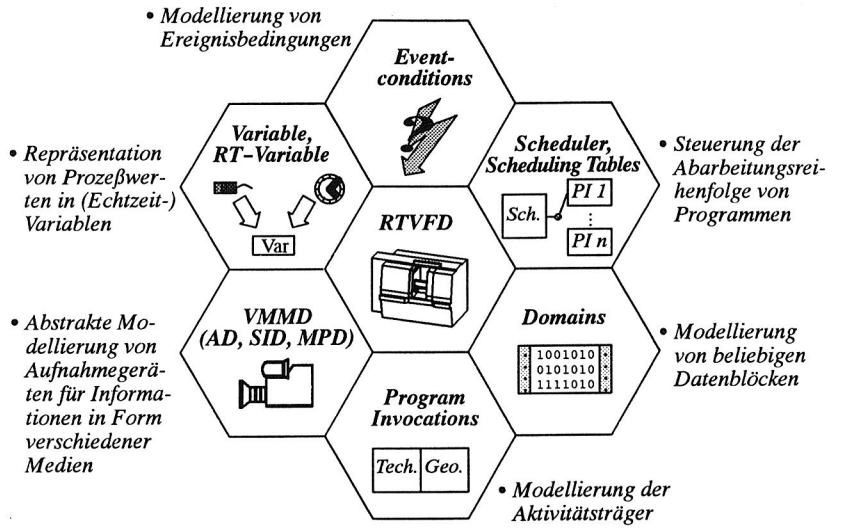


Bild 67: Objekte und ihre Aufgaben in RTSIMS

tung der Clients sowie zur Realisierung eines gegenseitigen Ausschlusses bezüglich des Zugriffs auf die wesentlichen Kontrollfunktionen der VMMDs berücksichtigt. Zusätzlich ist zur Vereinfachung der Definition der Objekttypen der Objekttyp „Generic Object" definiert. Er beinhaltet Basisdefinitionen für verschiedene andere Objekttypen, die an diese vererbt werden. Die Instantiierung von Objekten der zugehörigen Objektklasse ist nicht vorgesehen.

Eine nähere Spezifikation der Objekttypen und der auf sie anwendbaren Dienste erfolgt in den nachfolgenden Abschnitten. Eine Darstellung der Kardinalitäts- und Vererbungsbeziehungen für die Objekttypen ist in Bild 68 gegeben. Aus Gründen der Übersichtlichkeit wurde auf die Darstellung des „Generic Object" und dessen Vererbungsbeziehungen verzichtet.

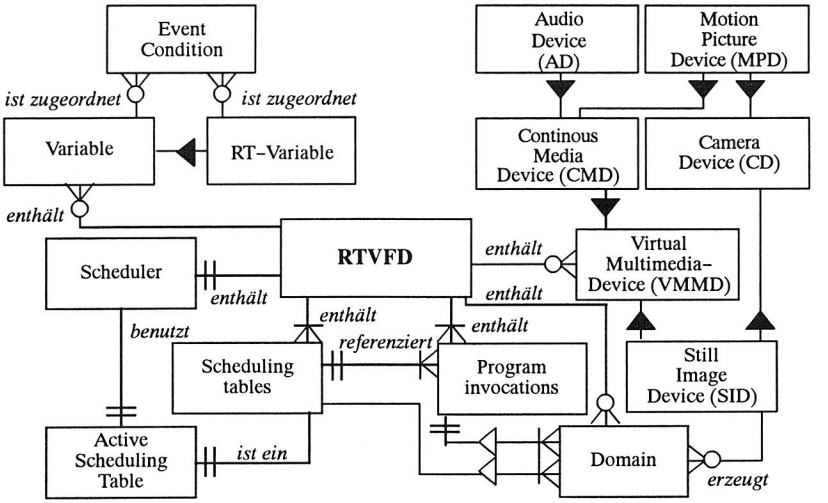
6.2.4 Generisches Objektmanagement

Den Ausführungen zu den verschiedenen Objekten, mit denen realen Geräte, ihre Eigenschaften und ihr nach außen sichtbares Verhalten, modelliert werden können, wird die Darstellung des zur Verringerung des Definitions- und Realisierungsaufwandes als Basis festgelegten generischen Objektes vorangestellt.

Object-type = Generic Object

Key Attribute	:	Identifier
Attribute	:	Status
Service	:	Get-Attributes
Service	:	Status
Service	:	Create
Service	:	Delete

Die für dieses Objekt definierten Dienste können auf Instanzen verschiedener, abgeleiteter Objekttypen wirken (Polymorphismus [138]) und sind in der Dienstgruppe des *Generic Object Ma-*



Legende: *(von links nach rechts auszuwerten)*

beliebig viele	*ein oder kein*	*beinhaltet*
mindestens ein	*genau ein*	*ist Basisklasse*

Die Kardinalität bezieht sich jeweils auf das Objekt, dem das Kardinalitätssymbol am nächsten ist. Die Beschriftung bezeichnet ggf. den Typ der Beziehung. Beschriftungen oberhalb oder rechts einer Linie betreffen das Verhältnis des links bzw. oben dargestellten Objekts zu dem weiter rechts bzw. weiter unten.Entsprechendes gilt für Beschriftungen unterhalb bzw. links einer Linie.

Bild 68: Beziehungen der Objekte in RTSIMS

nagement zusammengefaßt. Jeder Dienst kann für unterschiedliche Objekttypen eine andere Methode und andere Parameter haben. Der letztgenannte Aspekt wird durch die Kennzeichnung eines Parameters als „Selection" bzw. „Conditional" bei der Beschreibung der Dienstparameter berücksichtigt. Bei der Darstellung der Dienstelemente werden die Parameter, die spezifisch für die einzelnen Objekte sind, aufgeführt. Allen Diensten sind die Bedeutungen der Parameter „Object Type" und „Object ID" gemeinsam, die jeweils den Typ und die Identifikation des betroffenen Objektes bezeichnen.

Mit Hilfe des bestätigten Dienstes `STATUS` kann ein Benutzer den Zustand eines RTVFD, einer Programminstanz, eines Schedulers oder eines VMMD erfragen. Die Dienstantwort enthält neben den objektspezifischen Parametern den Stand der lokalen Uhr des Responders bei der Dienstausführung („Time"), die Güte der Synchronisation zu diesem Zeitpunkt sowie optional anwendungsspezifischeZusatzinformationen. Die Parameter der Dienstelemente sind in Tabelle 15 dargestellt.

Mit dem bestätigten Dienst `GET-ATTRIBUTES` kann der Benutzer die Attribute einer Instanz der Objekttypen RTVFD, Domain, Program Invocation oder VMMD erfragen. Die Parameter der zu diesem Dienst gehörigen Dienstelemente sind in Tabelle 16 dargestellt. Die typabhängigen Informationen, mit denen die Eigenschaften des Objektes, auf das in der Dienstanforderung Bezug genommen wird, modelliert werden, werden vom Responder zusammengestellt, in der Dienstantwort übertragen und an den Benutzer übergeben.

Parametername	req	ind	rsp	cnf
Argument	**M**	**M(=)**		
Object Type	M	M(=)		
Object ID	M	M(=)		
Result(+)			**S**	**S(=)**
Time			M	M(=)
Synchronisation Quality			M	M(=)
RTVFD Specific			S	S(=)
RTVFD Logical Status			M	M(=)
RTVFD Physical Status			M	M(=)
Program Invocation Status			S	S(=)
Scheduler Status			S	S(=)
VMMD Specific			S	S(=)
VMMD Operational Status			M	M(=)
Current Settings			M	M(=)
Current extended settings			C	C(=)
Local Detail			U	U(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 15: Parameter der Dienstelemente des Dienstes STATUS

Eine Sonderstellung nehmen die Attribute „List of Domain Reference", „List of Program Invocation Reference" und „List of Scheduling Table Reference" ein, da sie jeweils einen Verweis auf ein Domain enthalten, das zur Verwaltung der eigentlichen Nutzdaten verwendet wird. Eine beliebige Liste von dynamischen Objekten, z.B. die Liste der definierten Programminstanzen, ist in einem Domain als Folge von Objektbezeichnern gespeichert. Das RTVFD verwaltet diese Liste bei der Instantiierung neuer und dem Löschen vorhandener Objekte. Die Liste beinhaltet auch die

Parametername	req	ind	rsp	cnf
Argument	**M**	**M(=)**		
Object Type	M	M(=)		
Object ID	M	M(=)		
Result(+)			**S**	**S(=)**
RTSIMS Deletable			M	M(=)
RTVFD Description			S	S(=)
List of Domain Reference			C	C(=)
List of Program Invocations Reference			C	C(=)
List of Scheduling Tables Reference			C	C(=)
Domain Description			S	S(=)
Reference Count			M	M(=)
Program Invocation Description			S	S(=)
List Of Domain IDs			M	M(=)
VMMD Description			S	S(=)
Coding techniques			M	M(=)
Coding extension			M	M(=)
Movement range			M	M(=)
CD-Description			C	C(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 16: Parameter der Dienstelemente des Dienstes GET-ATTRIBUTES

Bezeichner aller statisch definierten Objekte eines Objekttyps. Die Verwaltung einer derartigen Liste ist dabei nur für die Objekttypen notwendig, bei denen auch dynamische Instanzen gebildet werden können. Die Objektbezeichner aller rein statisch instantiierten Objektinstanzen sind bereits in der Systemkonfiguration festgelegt. Der Zugriff auf die Liste der an einem RTVFD defi-

nierten Domains, Programminstanzen oder Schedulingtabellen ist mit den regulären Diensten zur Übertragung von Domains möglich.

Der bestätigte Dienst CREATE dient zur typunabhängigen Erzeugung von Objektinstanzen und kann auf Domains, Program Invocations und Scheduling Tables angewendet werden. Instanzen von Program Invocations und Schedulingtabellen werden aus Instanzen von Domains zusammengesetzt. Die hierfür benötigte Liste der Domains ist im Parameter „List of Domain IDs" enthalten (vgl. Tab. 17). Die Bestätigung enthält Informationen über den Erfolg der Dienstausführung.

Parameter Name	req	ind	rsp	cnf
Argument	**M**	**M(=)**		
Object Type	M	M(=)		
Object ID	M	M(=)		
List of Domain Names	C	C(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 17: Parameter der Dienstelemente des Dienstes Create

Der bestätigte Dienst DELETE dient zum typunabhängigen Löschen von dynamisch erzeugten Objektinstanzen. Die Parameter der zugehörigen Dienstelemente sind in Tabelle 18 dargestellt.

Parameter Name	req	ind	rsp	cnf
Argument	**M**	**M(=)**		
Object Type	M	M(=)		
Object ID	M	M(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Specification			M	M(=)

Tabelle 18: Parameter der Dienstelemente des Dienstes `Delete`

6.2.5 Das virtuelle, echtzeitfähige Feldgerät

Das zentrale Objekt des Servers ist das virtuelle Feldgerät mit Unterstützung von Echtzeitfunktionalität. Es dient zur Modellierung des nach außen sichtbaren Verhaltens eines Feldgerätes. Der Objekttyp des RTVFD ist wie folgt definiert:

Object-type = RTVFD
Generic Object :: Key Attribute : Object Identifier
Attribute : Vendor Name
Attribute : Model Name
Attribute : Revision
Attribute : Logical Status [State-changes-allowed, No-state-changes-allowed
Limited-services-permitted, Support-services-allowed]
Attribute : Physical Status [Operational, Partially-operational, Inoperable,
Needs-commissioning]
Attribute : List of Program Invocations
Attribute : List of Scheduling Tables
Attribute : List of Domains
Attribute : List of Upload-State-Machines

Attribute : List of VMMDs
Attribute : Scheduler reference
Generic Object :: Service : Status
Service : Unsolicited-Status
Service : Identify

Das Schlüsselattribut „Object Identifier" erlaubt eine eindeutige Referenzierung der Instanz des RTVFD-Objektes. Die Attribute „Vendor Name", „Model Name" und „Revision" enthalten Infomationen über den Hersteller und die Ausführung des Gerätes. Die „List of Program Invocations" verweist auf das Domain, in dem die Bezeichnungen aller am RTVFD definierten Instanzen der Objektklasse „Program Invocation" enthalten sind. Entsprechend referenzieren die Attribute „List of Scheduling Tables" die Liste der am RTVFD definierten Schedulingtabellen und „List of Domains" die Liste der vorhandenen Domains. Das Attribut „List of VMMDs" verweist auf die am RTVFD vorhandenen abstrakten Repräsentationen von Multimedia-Aufnahmegeräten. Für laufende Ladevorgänge von Domains vom Server an den Client („Upload") werden die zugehörigen Zustandsautomaten („Upload-State-Machines") in der „List of Upload-State-Machines" verwaltet. In RTSIMS erfolgt die Ausführung von Programminstanzen unter Kontrolle des Schedulers. Der Verweis auf die Instanz des Schedulers ist im Attribut „Scheduler reference" enthalten.

Der Dienst `Status` wird vom generischen Objekt geerbt. Der unbestätigte Dienst `UNSOLICITED-STATUS` dient zur spontanen Übermittlung der Statusinformationen eines RTVFD und wird von Serverapplikationen typischerweise dann benutzt, wenn ein Wechsel des Zustandes des RTVFD festgestellt wurde. Als Parameter der Dienstanforderung und -ankündigung werden die RTVFD-spezifischen Attribute der Dienstantwort des Dienstes `STATUS` genutzt.

Der bestätigte Dienst `IDENTIFY` dient zur Abfrage der Identifikation eines Servers. Die Dienstantwort enthält Parameter, mit deren Hilfe die Werte der diesbezüglichen Attribute („Vendor Name", „Model Name" und „Revision") übertragen werden.

6.2.6 Domain-Management

Domains sind Objekte, die beliebige Daten enthalten können. Sie können am Server statisch vorliegen oder dort autonom bzw. durch das Herunterladen vom Client aus neu definiert werden. Sofern sie zur Laufzeit des Systems angelegt wurden, können sich auch wieder gelöscht werden. Zudem besteht die Möglichkeit, Domains vom Server an den Client zu übertragen.

Object-type = Domain
Generic Object :: Key Attribute : Identifier
Attribute : Deletable
Attribute : Content
Attribute : Domain State [Loading, Ready, In-use]
Attribute : List of PIs
Attribute : List of Scheduling Tables
Attribute : Deleteable
Service : Initiate Download Sequence
Service : Download Segement
Service : Terminate Download Sequence
Service : Initiate Upload Sequence
Service : Upload Segment

Service : Terminate Upload Sequence
Service : Request Domain Upload
Service : Request Domain Download
Generic Object :: Service : Delete

Der Wert des Attributs „Deletable" zeigt an, ob ein existierendes Domain prinzipiell gelöscht werden kann. „Content" stellt einen Verweis auf die im Domain hinterlegten Daten dar. Ein Domain ist grundsätzlich in einem der Zustände „Loading", „Ready" oder „In-use". Während der dynamischen Erstellung befindet es sich im Zustand „Loading". Danach im Zustand „Ready". Ist ein Domain bei der Erstellung einer Programminstanz oder einer Schedulingtabelle verwendet worden, so befindet es sich im Zustand „In-use". In diesem Zustand kann es nicht gelöscht werden. Die Referenzen auf die Programminstanzen respektive auf die Schedulingtabellen, die unter Bezug auf ein spezifisches Domain erstellt wurde, werden in den Listen „List of PIs" bzw. „List of Scheduling-Tables" hinterlegt. Diese Listen werden bei der Erzeugung und Löschung von PIs und Schedulingtabellen aktualisiert.

Zum Transfer von Domains zwischen dem Client und dem Server sind zwei Dienstsequenzen definiert. Mittels der *Upload-Sequenz* können Domains vom Server zum Client geladen werden. Die *Download-Sequenz* dient zur Übertragung eines Domains in die andere Richtung. Die Dienstsequenzen und die Parameter der Dienste werden hier nicht näher erläutert. Sie entsprechen denen von FMS [111]. Die Gleichberechtigung aller Stationen in RTSIMS erlaubt allerdings die Übertragung von Domains zwischen allen Stationen in beiden Richtungen, wohingegen das Herunterladen eines Domains in FMS nur zwischen zwei Masterstationen möglich ist. Die Dienste `REQUEST-DOMAIN-UPLOAD` und `REQUEST-DOMAIN-DOWNLOAD` dienen der wechselseitigen Anforderung des Ladevorganges. Alle Dienste sind bestätigte Dienste. Zusätzlich zu den Dienstparametern ihrer Pendants in FMS beinhalten alle Dienste Parameter zur Spezifikation der Priorität der Datenübertragung und des Modus der Datenübertragung auf Ebene der Sicherungsschicht.

6.2.7 Prozeßvariable

Prozeßvariable, im nachfolgenden auch kurz als Variable bezeichnet, werden im Regelfall dazu benutzt, die zur Beschreibung des aktuellen Zustands eines technischen Prozesses wichtigen Zustandsgrößen zu modellieren. Sie stellen damit für die meisten Steuerungs- und Regelungsapplikationen die wichtigsten Kommunikationsobjekte dar. Der Austausch der Werte von Variablen ist anwendungsabhängig auf unterschiedliche Art erforderlich. Dieser Aspekt hat bereits bei der Konzeption der Protokolle der Sicherungsschicht Berücksichtigung gefunden. Die dort zur Verfügung gestellte Funktionalität für den periodischen und den aperiodischen Austausch von Variablen wird dem Benutzer als Dienstqualität der entsprechenden Dienste von RTSIMS angeboten.

Zur Modellierung einer Variable bzw. einer Echtzeitvariable (vgl. Kap. 2.4.1) existieren der Objekttyp „Variable" und der daraus abgeleitete Objekttyp „RT-Variable". Die gemeinsamen Operationen und Attribute aller Variablen sind in dem Objekttyp „Variable" enthalten, der die nachfolgend dargestellte Struktur hat:

Object-type = Variable
Generic Object:: Key Attribute : Identifier
Attribute : Variable Type
Attribute : Content
Attribute : Remote Validity

Service : Read
Service : Write

Instanzen vom Objekttyp Variable können den Prozeßwert („Content") einer Variablen vom Datentyp („Variable Type") zusammen mit der Kennung zur Anzeige von deren Ungültigkeit aus technischer Sicht („Remote Validity", vgl. Kap. 3.1) darstellen. Auf die Variablen kann mittels der Dienste `READ` und `WRITE` lesend bzw. schreibend zugegriffen werden. Variablen können in RTSIMS entweder einen einfachen oder einen zusammengesetzten Datentypen haben. Zu den einfachen Datentypen gehören (jeweils mit Angabe der Bezeichnung des Datentyps und der Größe):

- Wahrheitswerte (`boolean`: 1 Byte)
- Ganzzahlige Variable (`int8`, `uns8`: 1 Byte; `int16`, `uns16`: 2 Byte, `int32`, `uns32`, 4 Byte)
- Gleitkomma-Variable (`float`: 4 Byte)
- Zeichenketten (`VisibleString`: 1 bis 244 Byte) (`OctetString`: 1 bis 244 Byte)
- Bitketten (`BitString`: 8 bis 1952 Bit in 8-Bit Längenstufen)

Als Konstruktoren für zusammengesetzte Datentypen werden die Reihung (Array) und der Verbund (Record) angeboten. Zusammengesetzte Datentypen dürfen intern nicht weiter strukturiert sein.

Lesen einer Variablen

Die Leseoperation (`READ`) liefert atomar den aktuellen Wert einer Variablen und den Gültigkeitsstatus zurück. Die zu den Dienstelementen gehörigen Parameter sind in Tabelle 19 aufgeführt.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object ID	M	M(=)		
Read-mode	M			
Result(+)			**S**	**S(=)**
ID			M	M(=)
Value			M	C
Remote-Validity			M	C
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 19: Parameter der Dienstelemente des Dienstes `READ`

Um den unterschiedlichen Anforderungen der Anwendungen zu entsprechen, existieren für die Leseoperation drei verschiedene Ausführungsmodi, die durch Setzen eines Wertes für den Dienstgüteparameter „Read-mode" vom Dienstbenutzer ausgewählt werden können. Im *lokalen Modus* („local") wird der Wert der spezifizierten Variablen aus der lokalen Instanz der verteilten Datenbasis gelesen und an den Dienstbenutzer übergeben. Im *entfernten Modus* („remote") wird der Wert von der Instanz der verteilten Datenbasis der Erzeugerstation angefordert und erst nach dessen Eintreffen gelesen. Im Modus „*update*" wird eine Aufforderung zur Aktualisierung der Variablen über die RTSIMS-Protokollinstanz der produzierenden Station an die dortige Applikation übergeben. Von dieser wird mit der Antwort eine Aktualisierung der angeforderten Variable durchgeführt. Trifft in diesem Modus bei der RTSIMS-Instanz des Dienstinitiators die Antwort ein, so wird nach dem Eintreffen der nächsten Dienstankündigung, die eine Aktualisierung des Wertes in der Datenbank anzeigt, ein lokales Lesen initiiert. Der dabei erhaltene Wert wird im Dienstele-

ment der Dienstbestätigung an den Benutzer übergeben. Dieser Vorgang ist in Bild 69 veranschaulicht. Die Dienstelemente der Antwort und der Bestätigung werden nur im Modus „update" benötigt. Die Angabe eines Wertes für den Parameter „Transfer-mode" ist ebenfalls nur in diesem Mo-

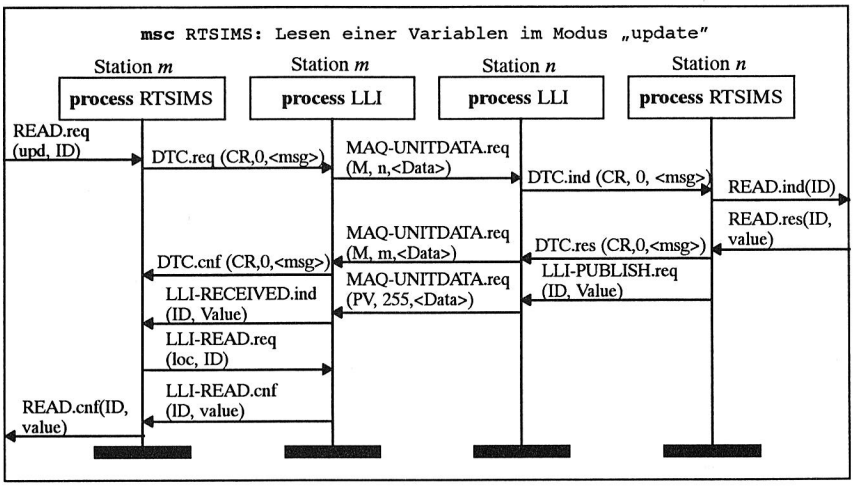


Bild 69: Lesen einer Variablen im Modus „update"

dus erforderlich. Da aus der Kennung der Variablen zwar die Adresse der produzierenden Station, nicht aber die Identifikation der sie dort erzeugenden Applikation hervorgeht, wird der Dienst über eine Kommunikationsbeziehung an den Dienstzugangspunkt mit der Kennung „15" gesendet und in der Zielstation an alle dort vorhandenen Applikationen übergeben. Der Parameter „Object ID" beinhaltet den Bezeichner der Variablen. Über diesen kann aus der lokalen MIB auch der Datentyp der Variablen festgestellt werden. In den Modi „update" und „remote" ist zusätzlich die Festlegung einer Priorität für die Dienstanforderung erforderlich. Das Dienstelement der Bestätigung wird auch zur Anzeige der erfolgten Aktualisierung einer konsumierten Variable verwendet. In diesem Fall wird nur die Kennung der Variablen, nicht aber deren Wert übergeben.

Schreiben einer Variablen

Die Aktualisierung eines Variablenwertes durch die erzeugende Applikation erfolgt mittels des Dienstes `WRITE`. Für diesen Dienst sind, wie in Tabelle 20 dargestellt, nur die Dienstelemente der Anforderung und der lokalen Ankündigung notwendig.

Parameter Name	req	cnf
Argument	M	
Object ID	M	M
Value	M	
Remote-Validity	M	
Write-mode	M	

Tabelle 20: Parameter der Dienstelemente des Dienstes WRITE

Mittels des Parameters „Object ID" wird die ausgewählte Variable eindeutig referenziert. Ihr Wert wird im Parameter „Value" übergeben. Der Parameter „Remote Validity" wird entsprechend der

festgestellten technischen Gültigkeit des Wertes der Variablen gesetzt. Durch Wahl eines Wertes für den Parameter „Write-mode" kann festgelegt werden, ob die Aktualisierung nur lokal erfolgen oder auch die verteilten Kopien betreffen soll. In letzterem Fall kann über den Parameter „Priority" die Priorität des Datenaustausches festgelegt werden. Die lokale Bestätigung wird in einer produzierenden Station nach dem Austausch der Variable über die verteilte Datenbasis erzeugt.

Besonderheiten der Echtzeitvariable

Der Objekttyp „RT-Variable" erbt vom Objekttyp „Variable" dessen Attribute und den Dienst `READ`. Zusätzlich existieren die Operationen `RT-WRITE` und `RT-READ`, die das Setzen bzw. die Auswertung der besonderen Attribute von Echtzeitvariablen vornehmen. Dabei handelt es sich um die Behandlung der Gültigkeitsdauer („Validity Period") des Variablenwertes ab dem Zeitpunkt der Gültigkeit („Valid from") unter Berücksichtigung der Synchronisationsgüte der lokalen Station beim Schreiben und der entfernten Station beim Lesen. Ob es sich bei dem frühesten Zeitpunkt der Gültigkeit um den Erzeugungszeitpunkt oder um einen beliebigen, durch die Applikation gewählten, Zeitpunkt handelt, wird durch den Wert des Attributes „Produced At" angezeigt. Im Vergleich zu den Konzepten von MPS-E in OLCHFA, die nur eine Zeitstempelung mit dem Wert des Zeitpunktes der Erzeugung zulassen, besteht hier damit zusätzlich die Möglichkeit, durch Wahl eines in der Zukunft liegenden Zeitpunktes für die beginnende Gültigkeit eines Wertes, eine Synchronisation der verteilten Applikationen vorzunehmen.

Object-type = Variable :: RT-Variable
Variable :: Key Attribute : Identifier
Variable :: Attribute : Variable Name
Variable :: Attribute : Content
Variable :: Attribute : Remote Validity
Attribute : Validity Period
Attribute : Valid from
Attribute : Produced At [Release Time, Other]
Attribute : Synchronisation Quality
Variable :: Service : Read
Service : RT-Read
Service : RT-Write

Der Gültigkeitszeitpunkt wird durch eine Zeitangabe im Format der Systemzeit gebildet. In OSIRIS ist dies ein ganzzahliger, vorzeichenloser Wert aus dem Bereich von 0 bis $2^{32}-1$, der für die interne Darstellung und die Übertragung auf eine Bitfolge der Länge 32 abgebildet werden kann. Die Gültigkeitszeitdauer ist ebenfalls ein ganzzahliger, vorzeichenloser Wert, der sich auf den gleichen, von der Systemkonfiguration abhängigen, Zeitmaßstab (vgl. Kap. 7.6) bezieht. Die möglichen Werte sind aufgrund der Beschränkung auf eine Bitfolge der Länge 28 aus dem Bereich zwischen 0 und $2^{28}-1$. Die Werte des Attributes „Produced At" können mittels eines Bits modelliert werden. Für die Güte der Synchronisation sind drei Qualitätsstufen reserviert, die in zwei Bit verschlüsselt sind. Dabei bedeutet ein hoher Wert eine hohe Synchronisationsgüte. Der Wert „0" zeigt an, daß die Station nicht synchronisiert ist. Der Umfang der Parameter, die sich auf die besonderen Aspekte einer Echtzeitvariablen beziehen, beträgt somit acht Byte. Die maximale Länge der weiteren Nutzdaten verkürzt sich entsprechend.

Der lesende Zugriff auf zeitkritische Variable erfolgt mittels des Dienstes `RT-READ`. Die zu den Dienstelementen gehörigen Parameter sind in Tabelle 21 dargestellt. Die Bedeutung des Parame-

ters „Read-mode" entspricht der, die dieser Parameter beim Dienst `Read` hat. Ist die Gültigkeit der Variablen noch nicht abgelaufen, und war die Synchronisationsgüte bei der Erzeugerstation zum Zeitpunkt der Erzeugung ausreichend gut, so wird der Prozeßwert samt der Werte der zeitkritischen Attribute an den Benutzer übergeben. Sollte die Bedingungen für eine temporale Gültigkeit nicht mehr erfüllt sein, so wird ein Fehlermeldung an den Dienstbenutzer übergeben. Der Prozeßwert wird zusammen mit den Attributen und einer entsprechenden Warnung an den Dienstbenutzer übergeben, wenn entweder die zeitliche Gültigkeit der Variablen noch nicht erfüllt ist, oder die Synchronisationsgüte in der Erzeuger- oder der lokalen Station nicht ausreichend gut war, bzw. ist.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M**		
Object ID	M	M(=)		
Read-mode	M			
Result(+)			**S**	**S(=)**
ID			M	M(=)
Value			M	C
Remote-Validity			M	C
Validity Period			M	C
Valid from			M	C
Produced at			M	C
Warning			C	C(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 21: Parameter der Dienstelemente des Dienstes RT-READ

Der reguläre Dienst `READ` kann ebenfalls auf RT-Variable angewendet werden (Polymorphismus). Dabei werden die Aspekte der temporalen Gültigkeit der Variable nicht betrachtet, sondern nur deren Wert zurückgeliefert. Eine beispielhaft Veranschaulichung des Polymorphismus ist in Bild 70 dargestellt.

Der zeitkritische Dienst `RT-WRITE` dient zur Aktualisierung des Wertes einer zeitkritischen Variable in deren Erzeugerstation. Sofern die Gültigkeitsdauer des Wertes noch nicht abgelaufen ist, wird dieser über das LLI an die lokale Instanz von DRSLP übergeben. Anderenfalls wird eine Fehlermeldung an den Benutzer übergeben und die Anforderung wird nicht an die Instanzen der unterlagerten Protokolle weitergegeben. Je nach Wert des Attributes „Write-mode" erfolgt die Aktualisierung entweder nur in der lokalen Station, oder es wird durch den Dienstaufruf auch eine Aktualisierung der verteilten Kopien bewirkt.

6.2.8 Verarbeitung von Ereignissen

Die Ereignisverarbeitung von RTSIMS basiert auf dem bereits für das OLCHFA-System eingeführten Prinzip, daß Variablen Grenzwerte zugewiesen werden können, bezüglich deren Über- bzw. Unterschreitung sie dann bei schreibenden Zugriffen in der produzierenden Station überwacht werden. Es sind wiederum die drei Ereignisklassen *Ascending Threshold*, *Descending Threshold* und *Deadband* definiert (vgl. Bild 71). Bezüglich der Verletzung von Grenzwerten können dabei Variablen oder Echtzeitvariablen vom Datentyp „Boolean", beliebige ganzzahlige Typen sowie Gleitkommazahlen überwacht werden.

Der Objekttyp der Ereignisbedingung (Event Condition) ist wie nachfolgend dargestellt definiert:

Object-type =Event-Condition
Generic Object :: Key Attribute : Identifier

Attribute : Referenced Variable
Attribute : Event-ID
Attribute : Max-Sending delay
Attribute : Reference Value
Attribute : Event type [Ascending, Descending, Deadband]
Constraint : Event type = Deadband
Attribute : Bandwidth

Der Aufbau dieses Objekttyps erlaubt die Referenzierung einer Variable, die durch die Ereignisverarbeitung überwacht wird, durch den Wert des Attributs „Referenced Variable". Bezüglich einer Variable können mehrere, voneinander unabhängige Ereignisbedingungen definiert sein. Eine eindeutige Kennung der Ereignisbedingung wird mittels des Parameters „Event-ID" erreicht. Das Attribut „Max-Sending delay" dient zur Modellierung der maximal tolerablen Sendeverzögerung. Im Attribut „Reference Value" wird der Grenzwert für die Variablen hinterlegt, bezüglich dessen Über- bzw. Unterschreitung sie überwacht wird. Ein Ereignis wird in Abhängigkeit vom Ereignistyp („Event type") genau dann erkannt, wenn der Wert der Variablen den Grenzwert überschreitet, unterschreitet oder sich nicht mehr innerhalb der erlaubten Schwankungsbandbreite („Bandwidth") um den Referenzwert befindet. Die verwendeten Ordnungsrelationen sind „$\leq$" und „$\geq$" für ganzzahlige und Gleitkomma-Variablen. Diese erlauben eine lineare Ordnung auf der jeweiligen Zahlenmenge, d.h. sie sind reflexiv, transitiv, antisymmetrisch und erlauben Vergleichbarkeit [139]. Boolesche Variable können, mit der allgemein üblichen Definition des Zahlenwerte „1" zur Repräsentation des logischen Wertes „Wahr" und „0" für „Falsch", wie ganzzahlige Variable mit eingeschränktem Wertebereich behandelt werden. Die Belegung der Attribute mit Werten erfolgt

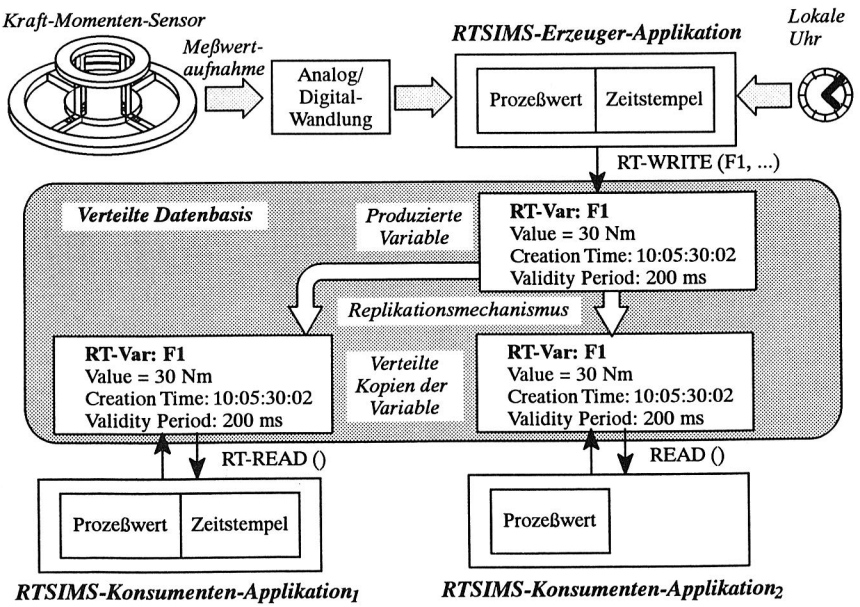


Bild 70: Polymorphismus am Beispiel einer zeitkritischen Variable

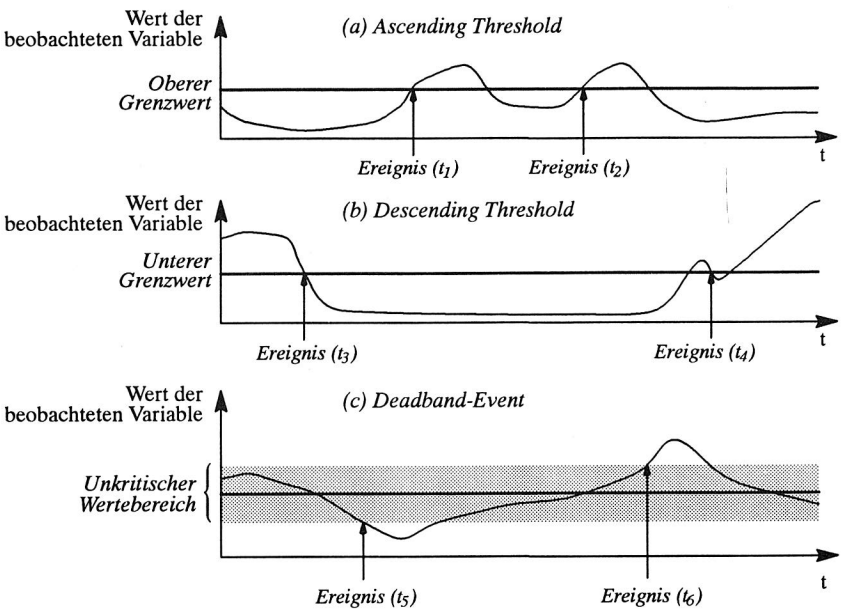


Bild 71: Berücksichtigte Ereignisklassen

während der Konfigurierungsphase des Systems und ist in der lokalen MIB der Station hinterlegt, die als Produzent der bezogenen Variable auftritt.

Zur Laufzeit des Systems wird bei jedem schreibenden Zugriff auf eine Variable in der produzierenden Station überprüft, ob eine oder mehrere Ereignisbedingungen erfüllt sind. Ist dies der Fall, so wird pro erfüllter Ereignisbedingung eine diesbezügliche Benachrichtigung abgesetzt. Zu diesem Zweck ist der Dienst `EVENT-NOTIFICATION` definiert. An der Benutzerschnittstelle steht nur das Dienstelement der Ankündigung zur Verfügung, dessen Parameter in Tabelle 22 dargestellt sind. Der Aufruf wird durch die Ereignisverarbeitung selbsttätig zusammengestellt und im hochprioren, ungesicherten Modus unter Berücksichtigung der maximalen Sendeverzögerung abgesetzt. Ist die Ausführung des Dienstes durch das Kommunikationssystem nicht möglich, erfolgt eine unverzügliche Meldung an die Applikation mittels des Dienstes `REJECT`. Diese kann die dann notwendigen lokalen Maßnahmen zur Behandlung des eingetretenen Ereignisses, z.B. den sofortigen Stopp des Prozesses, treffen.

Parameter Name	ind
Argument	**M**
Variable-ID	M
Event-ID	M
Time	M
Synchronisation Quality	M

Tabelle 22: Parameter der Dienstelemente des Dienstes EVENT-NOTIFICATION

Als Parameter der Dienstelemente des Dienstes `EVENT-NOTIFICATION` werden die Identifikation der Variablen, die Identifikation des Ereignisses, der Zeitpunkt der Feststellung des Ereignis-

ses und die Güte der Uhrzeitsynchronisation in der produzierenden Station zu diesem Zeitpunkt übermittelt (vgl. Tab. 22). Handelt es sich um eine Echtzeitvariable, so werden dabei deren Erzeugungszeitpunkt und die festgestellte Synchronisationsgüte verwendet, anderenfalls die durch die Ereignisverwaltung direkt von der lokalen Uhr abgefragten Werte.

Es kommt im Bereich der Ereignisverarbeitung wiederum zu einem engen Zusammenhang zwischen der nachrichtenorientierten Kommunikation und der, die über die verteilte Datenbasis abgewickelt wird. Diesem Umstand muß in zweifacher Hinsicht Sorge getragen werden: Zum einen muß im Falle eines nur lokalen Schreibzugriffs der produzierenden Applikation die Ereignisverarbeitung für eine Aktualisierung der verteilten Kopien sorgen, und zum anderen muß die Mitteilung über den Eintritt des Ereignisses an alle Stationen gesendet werden, da beim Produzenten keine Informationen darüber vorliegen, welche Applikationen als Konsumenten der Variablen auftreten. Diese Mitteilung wird damit auch der lokalen Applikation zugestellt. Sie kann über eine beliebige Kommunikationsbeziehung außer der, die für das Management reserviert ist, abgesetzt werden.

Da nicht in jeder Station zu jeder Zeit eine verbindungslose Kommunikationsbeziehung vorhanden ist, hat der Mechanismus der Ereignisverarbeitung dafür Sorge zu tragen, daß bei der Initialisierung der Station mindestens eine derartige Verbindung aufgebaut wird.

Die in den Stationen nach Eintreffen einer Benachrichtigung über den Eintritt eines Ereignisses zu ergreifenden Maßnahmen können mittels entsprechender Einträge in der Schedulingtabelle spezifiziert werden (vgl. Kap. 6.2.10).

6.2.9 Programminstanzen

Programminstanzen dienen als Aktivitätsträger [140] des RTVFD. Ihre Ausführung wird durch den Scheduler gesteuert. Für die Erzeugung und das Löschen von dynamischen Programminstanzen werden die Dienste `CREATE` und `DELETE` mit entsprechend belegten Parametern genutzt.

Object-type = Program Invocation
Generic Object :: Key Attribute : ID
Attribute : Status [LOADING, IDLE, RUNNING]
Attribute : List of Table-References
Generic Object :: Service : Status
Generic Object :: Service : Create
Generic Object :: Service : Delete

Während der Ausführung einer Programminstanz (Zustand „Running") werden zu deren Verwaltung drei weitere Zustände, „Bereit", „Laufend" und „Unterbrochen", unterschieden, die jedoch nur intern von weiterer Bedeutung sind.

6.2.10 Steuerung der Ablaufreihenfolge von Programminstanzen

Die Ausführung von Programmen zur Steuerung eines realen Fertigungsgerätes obliegt in RTSIMS nicht wie in MMS oder FMS direkt dem Anwender, sondern der lokalen Instanz des Objektes zur Steuerung des Ablaufs [140], die hier, dem allgemeinen Sprachgebrauch folgend, als *Scheduler* bezeichnet wird. Der Scheduler startet und stoppt Programminstanzen gemäß der in der Schedulingtabelle vorliegenden Informationen. Programminstanzen können dabei zeit- oder ereignisgesteuert und verdrängend (*preemptiv*) oder nicht-verdrängend (*non-preemptiv*) [141] gestartet werden. Die Zusammenhänge zwischen Scheduler, der Schedulingtabelle und den Programminstanzen sind in Bild 72 dargestellt. Die Prioritätensteuerung sorgt dafür, daß laufende

Programminstanzen nur durch Programminstanzen höherer Priorität verdrängt werden können. Für die Prioritäten sind Werte zwischen 0 und 9 vorgesehen, wobei ein niedriger Wert eine niedrige Priorität impliziert. Nicht aktive Prozesse können keine Ressourcen halten. Alle Systemressourcen sind stets für den aktiven Prozeß verfügbar, wodurch eine Verklemmungsfreiheit erreicht wird [141].

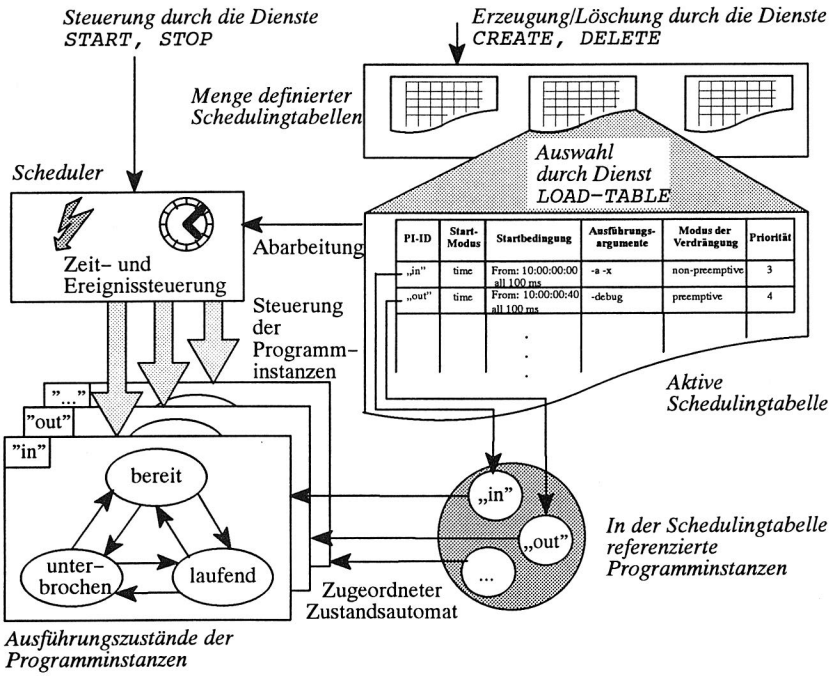


Bild 72: Scheduling von Programminstanzen: Darstellung der Zusammenhänge

Eine Instanz eines Scheduler-Objektes enthält Attribute zur Modellierung ihres aktuellen Zustandes, Referenzen auf die verfügbaren Schedulingtabellen und einen Verweis auf die Schedulingtabelle, die aktuell genutzt wird.

Object-type = Scheduler
Generic Object :: Key Attribute : ID
Attribute : Status [EMPTY, LOADING, STOPPED, STOPPING, PREPARING, RUNNING]
Attribute : List of Table-References
Attribute : Active table
Generic Object :: Service : Status
Service : Load-Table
Service : Start
Service : Stop

Zur Steuerung des Schedulers sind drei bestätigte Dienste definiert. Diese erlauben das Laden einer Schedulingtabelle (LOAD-TABLE), sowie den Start (START) und die Beendigung (STOP) von deren Abarbeitung. Nach dem Empfang der Anforderung zum Laden einer Schedulingtabelle wird die entsprechende Aktion durch die Server-Applikation ausgeführt, wenn noch keine Schedulingtabelle geladen ist oder der Scheduler sich im Zustand „Stopped" befindet. Die Dienstantwort bzw. -bestätigung enthält Informationen über den Erfolg der Ausführung des Dienstes. Die Parameter der Dienstelemente sind in Tabelle 23 aufgeführt.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 23: Parameter der Dienstelemente des Dienstes LOAD-TABLE

Die Parameter der Dienstelemente der Dienste START und STOP sind für beide Dienste identisch und in Tabelle 24 dargestellt.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 24: Parameter der Dienstelemente der Dienste START und STOP

Die Dienstbestätigung enthält im Fehlerfall eine Spezifikation des aufgetretenen Problems. Nach Eintreffen einer Anforderung zum Beginn der Abarbeitung einer Schedulingtabelle wird diese so lange bearbeitet, bis explizit eine Beendigung der Bearbeitung angefordert wird.

Die Zustandsübergänge des Scheduler-Objektes nach dem Empfang von Dienstanforderungen bzw. nach dem Versenden von Dienstanworten sind in Bild 73 in einem entsprechenden Diagramm dargestellt. Beim Server eintreffenden Dienstanforderungen ist in der Darstellung ein „+", den vom Server gesendeten Dienstantworten ein „–" vorangestellt. Die Zwischenzustände „Loading", „Preparing" und „ Stopping" werden nach Eintreffen einer Dienstanforderung nur kurzzeitig bis zum Senden der zugehörigen Antwort eingenommen.

Eine Schedulingtabelle besteht neben den Attributen zur Aufnahme der Identifiktion und zur Modellierung des Zustandes aus einer Liste von einem oder mehreren Scheduling-Einträgen (Scheduling-Entry).

Object-type = Scheduling-Table
Generic Object :: Key Attribute : ID
Attribute : Status [IDLE, IN-USE]
Attribute : List of Scheduling Entries
Generic Object :: Service : Create
Generic Object :: Service : Delete

Schedulingtabellen können vordefiniert sein oder zur Laufzeit des Systems durch den Dienst CREATE instantiiert werden. Hierfür werden die entsprechenden Daten als Parameter der

CREATE-Dienstankündigung übergeben. Sind alle Domains vorhanden und existiert noch keine Schedulingtabelle mit der spezifizierten Identifikation, so wird die Schedulingtabelle erzeugt und eine positive Quittung zurückgesendet. Ansonsten erfolgt eine negative Quittierung mit einer Spezifikation des aufgetretenen Fehlers. Das Löschen einer Schedulingtabelle mittels des Dienstes DELETE ist nur möglich, wenn diese sich nicht im Zustand „IN-USE" befindet, den sie immer dann einnimmt, wenn sie vom Scheduler bearbeitet wird.

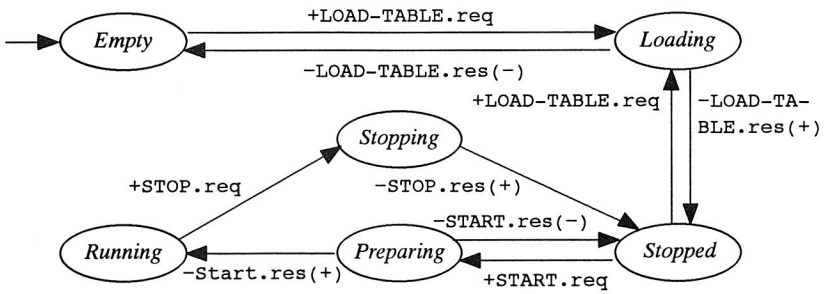


Bild 73: Zustandsübergänge des Schedulers

Ein einzelner Listeneintrag in der Schedulingtabelle beinhaltet folgende Bestandteile:

- *PI-ID:* Eindeutige Referenzierung der Programminstanz
- *Start-Modus:* Angabe, ob es sich um eine zeit- oder ereignisgesteuerte Einplanung der Programminstanz handelt
- *Startbedingungen:* Startbedingungen für die Programminstanz. Diese sind abhängig vom Startmodus. Für ereignisgesteuerte Einplanung erfolgt hier die Spezifikation der Ereignisse, auf deren Eintreten hin die Programminstanz gestartet werden soll. Bei zeitgesteuerter Einplanung besteht die Möglichkeit, die Programminstanz entweder einmalig oder zyklisch ab einem bestimmten Zeitpunkt mit einer gegebenen Periodizität einzuplanen.
- *Ausführungsargument:* Textuelle Information, die beim Start einer Programminstanz an diese übergeben wird.
- *Priorität:* Die statische Festlegung einer Priorität erlaubt die Steuerung der Einplanung der Programminstanzen nach einer benutzerdefinierten Gewichtung.
- *Modus der Verdrängung:* Ist bei Erfüllung der Startbedingung einer Programminstanz bereits eine andere Programminstanz in Bearbeitung, so wird gemäß des Wertes dieses Parameters entweder die laufende Programminstanz zu Ende bearbeitet (*non-preemptiv*) oder unterbrochen (*preemptiv*).

Eine zeitgesteuerte Einplanung von Programminstanzen wird nur dann ausgeführt, wenn die lokale Uhr der Station ausreichend genau synchronisiert ist. Ereignisgesteuerte Einplanungen werden unabhängig davon vorgenommen.

Unterbrochene Abarbeitungsvorgänge von Programminstanzen werden nach Beendigung der Programminstanz, durch deren Einplanung sie unterbrochen wurden, wieder eingeplant. Während der Ausführung von Programminstanzen bereit werdende andere Programminstanzen niedrigerer und gleicher Priorität sowie nicht-verdrängende Programminstanzen höherer Priorität werden in

einer Warteschlange nach Priorität, und innerhalb derer nach Startzeitpunkten geordnet, zwischengespeichert.

Eine beispielhafte Schedulingtabelle ist in Bild 74 dargestellt. Darin werden zwei zeitgesteuerte Programminstanzen und drei ereignisgesteuerte Instanzen, denen der Deutlichkeit halber symbolische Bezeichnungen zugewiesen sind, durch den Scheduler verwaltet. Die zeitgesteuerten Programminstanzen sorgen für eine Aufnahme von Prozeßwerten, deren Analog/Digitalwandlung

PI-ID	Start-Modus	Startbedingung	Ausführungs-argumente	Modus der Verdrängung	Priorität
„Eingabe"	time	From: 10:00:00:00 all 100 ms	„-a -x"	non-preemptive	3
„Ausgabe"	time	From: 10:00:00:40 all 100 ms	„-debug"	preemptive	4
„Notaus"	event	Variable: 0 1, Event: 3		non-preemptive	1
„Stufe-A"	event	Variable: 2 1, Event: 1 Variable: 15 9, Event: 7		preemptive	9
„Stufe-B"	event	Variable: 24 4, Event: 1	„-archive"	non-preemptive	1
. . .					

Bild 74: Beispielhafte Schedulingtabelle

(„Eingabe"), sowie für die Übergabe von Signalen an die Prozeßperipherie („Ausgabe"). Die ereignisgesteuerten Programminstanzen werden nach Eintreffen externer Ereignismeldungen aufgerufen. Der Programminstanz „Notaus" wird dabei die höchste Priorität zugewiesen. Sie wird nach dem Eintreffen einer Benachrichtigung über den Eintritt des Ereignisses, das von der Station „0" bezüglich der Variablen „1" festgestellt wird und dem die Identifikation „3" zugewiesen ist, aufgerufen. Die anderen ereignisgesteuerten Programminstanzen besitzen niedrigere Priorität und werden nur dann aufgerufen, wenn keine andere Programminstanz aktiv ist.

In jeder Schedulingtabelle ist implizit die Programminstanz „idle" enthalten, die immer dann aufgerufen wird, wenn keine andere Programminstanz eingeplant ist und die durch jede andere Programminstanz zu jeder Zeit verdrängt werden kann.

Der Mechanismus des Schedulings von Programminstanzen stellt, verglichen mit dem Verfahren wie es in FMS zur Steuerung der Ausführung von Programminstanzen Anwendung findet, eine funktionale Erweiterung dar. Das Verfahren von FMS kann als Sonderfall modelliert werden, indem pro Programminstanz von FMS eine Schedulingtabelle von OSIRIS mit einem einzigen Eintrag einer immer laufenden Programminstanz vorgesehen wird, deren Ausführung durch Laden und Start der Abarbeitung der entsprechenden Schedulingtabelle vorgenommen wird. Die Möglichkeit, mehrere Einträge in einer Schedulingtabelle und mehrere, dynamisch erzeugbare Schedulingtabellen pro RTVFD vorzusehen, führt jedoch einerseits zu einer weitaus größeren Flexibilität und erlaubt andererseits eine Strukturierung der Applikation auf einer hohen Abstraktionsebene. Auf dieser Ebene können die problemlösenden Bestandteile der Applikationen, z.B. die Ansteuerung von Prozeßperipherie, getrennt von deren Ausführungsbedingungen modelliert werden. Diese Strukturierung und die Parameterisierung der Ausführungsbedingungen über die Schedulingtabelle ermöglichen auch eine einfache Wiederverwendbarkeit bereits definierter Programminstanzen.

In einer Implementierung werden die benötigten Basismechanismen im Regelfall durch die verwendeten Echtzeitbetriebssysteme, wie z.B. *pSOS(+)*, *QNX* oder durch die Programmiersprache *PEARL* mit dem zugehörigen Betriebssystem [142], erbracht. Der Scheduler stellt somit für die in anderen Stationen laufenden Applikationen eine definierte, über das Kommunikationssystem verfügbare Schnittstelle zu diesen Mechanismen dar.

6.2.11 Unterstützung von Multimedia-Diensten und -Objekten

Im diesem Abschnitt werden die Objekttypen und die dazugehörigen Operationen zur Unterstützung von verteilten Multimedia-Applikationen eingeführt. Die Objekte dienen dabei als abstrakte Modelle von realen Geräten bzw. Systemen, die es erlauben, Bewegtbilder, Standbilder oder Audio-Daten zu generieren. Diese Abstraktion gestattet es, die real vorhandenen Geräte unabhängig von ihrer Ausprägung und ihren Spezifika auf eine standardisierte Weise anzusprechen und zu steuern. Die Vererbungshierarchie für Multimedia-Geräte ist in Bild 68 dargestellt. Dabei dient das *VMMD* zur Repräsentation der allen Systemen gemeinsamen Attribute und Operationen. Der Objekttyp Kamera (Camera Device, CD) dient der Modellierung der Gemeinsamkeiten optischer Aufnahmegeräte und der des CMD (Continous Media Device) für Geräte, die kontinuierliche Datenströme erzeugen. Reale Aufnahmegeräte können als Audio-Aufnahmegeräte (AD, Audio Device), Stand- (Still Image Device, SID) oder Bewegtbildaufnahmegeräte (Motion Picture Device, MPD) modelliert werden. Dabei kann ein reales Gerät durchaus auf verschiedene Objekttypen abgebildet werden. Eine hochwertige Kamera verfügt beispielsweise oftmals über ein eingebautes bzw. angeschlossenes Mikrofon und ist in der Lage, selbständig oder in Verbindung mit einer Bildaufnahmebaugruppe und deren Software, neben Bewegtbildern auch Standbilder zu erzeugen. Damit ließe sie sich als AD, SID oder MPD modellieren.

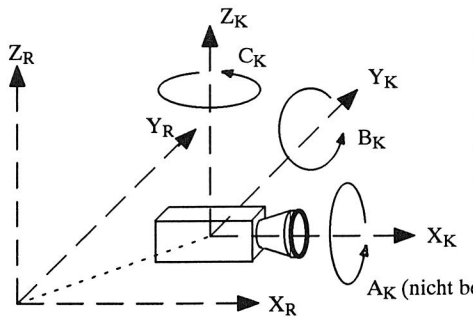
Das VMMD beinhaltet einige Parameter und Operationen, die es vom generischen Objekt erbt.

Objekttyp = VMMD
Generic Object :: Key Attribute : ID
Attribute : State [OPERATIONAL, BUSY, NOT ACCESSIBLE]
Attribute : List of supported coding techniques
Attribute : Coding type extension
Attribute : Movement-range
Attribute : List of Speed-profiles
Attribute : Current-settings
Attribute : Controlling-client
Generic Object :: Service : Get Attributes
Generic Object :: Service : Status
Service : Set
Service : Access-Control
Service : Coder-Control

Das Attribut „State" dient zur Aufnahme des Status des Gerätes. Die von dem Gerät unterstützten Kodierungstechniken, z.B. „H.261" und „MPEG-1" werden in dem Attribut „List of supported coding techniques" hinterlegt. Die unterstützten Parameterbereiche werden für jedes Kodierungsverfahren im Parameter „Coding type extension" geführt.

Zur Modellierung des Verfahrbereiches eines nicht ortsfesten Aufnahmesystems, dies kann z.B. eine Kamera sein, die an einem Handhabungsgerät montiert ist, dient der Parameter „Movement-

range". Für die betrachteten Typen von Aufnahmegeräten ist dabei eine Beweglichkeit des Systems mit maximal fünf Freiheitsgraden ausreichend. Die mögliche rotatorische Bewegung um die X_K-Achse des Körperkoordinatensystems des VMMD (vgl. Bild 6.8 nach *Raab* [143]) ist bei ADs nicht notwendig und kann bei den durch SIDs oder MPDs erzeugten Informationen durch Mechanismen der Datenverarbeitung nachgebildet werden.



Beschreibung einer Position durch den Orts- und Orientierungsvektor
$Pos = (X_R, Y_R, Z_R, B_K, C_K)$
relativ zum Ursprung des Referenzkoordinatensystems R bzw. des Körperkoordinatensystems K

Bild 75: Kartesisches Körperkoordinatensystem eines VMMD mit Orientierungswinkeln

Die Angabe einer Position und Orientierung des Körperkoordinatensystems K erfolgt relativ zu einem ortsfesten, kartesischen Referenzkoordinatensystem R, das z.B. durch das Maschinenkoordinatensystem gegeben sein kann. Eine Bewegung kann durch die Elementartransaktion [144] $P = \{T_{XR}, T_{YR}, T_{ZR}, R_{BK}, R_{CK}\}$ beschrieben werden. Dabei bezeichnen T_{XR}, T_{YR}, T_{ZR}, translatorische Bewegungen entlang der X_R-, Y_R- bzw. Z_R-Achse des Referenzkoordinatensystems und R_{BK}, R_{CK} Rotationen um die Y- bzw. die Z-Achse des Körperkoordinatensystems. Eine Abbildung der Koordinatensysteme auf die Kinematik verschiedener Handhabungssysteme ist mittels der aus der Robotik bekannten Methoden möglich [145, 146]. Die möglichen Geschwindigkeitsprofile, d.h. Kombinationen von Beschleunigungs- und Bremsrampen sowie der maximalen Verfahrgeschwindigkeit, sind vordefiniert und im Parameter „List of Speed-profiles" hinterlegt. Dies dient insbesondere dem kontrollierten Schwenk einer Kamera, z.B. zur Verfolgung von bewegten Objekten, und der Vermeidung von Beschädigungen der Aufnahmegeräte durch zu hohe Beschleungigungskräfte.

Ist eine Bewegung des Gerätes möglich, oder handelt es sich um eine Kamera, deren besonderen Funktionen, z.B. die Verschlußzeit oder die verstellbare Brennweite, sich fernsteuern lassen, so ist es notwendig, eine Zugriffskontrolle bezüglich dieser Funktionen einzuführen, um zu verhindern, daß während der Datenaufnahme die Aufnahmesysteme derart verstellt werden, daß die aufgenommenen Daten für zumindest einen Teil der Applikation wertlos werden. Aus diesem Grunde wird ein Semaphor-Mechanismus bezüglich der Kontrollfunktionen eingeführt. Zu einem bestimmten Zeitpunkt ist jede Instanz eines VMMD bezüglich der Kontrolle der Bewegungen und ggf. weiterer Funktionen grundsätzlich maximal einer Client-Applikation zugeordnet. Zum Auf- und Abbau dieser Zuordnung wird der vom Client initiierte Dienst `ACCESS-CONTROL` verwendet. Die Parameter der Dienstelemente sind in Tabelle 25 dargestellt. Dabei wird durch den Wert des Parameters „mode" angezeigt, ob die Kontrolle erbeten oder freigegeben wird. Die positive Dienstbestätigung ist parameterlos, in der negativen Dienstbestätigung wird der Rückweisungsgrund angegeben.

Der Ablauf nach Empfang einer ACCESS-CONTROL-Dienstanforderung stellt sich im Server wie folgt dar: Ist die Instanz des VMMD bei Empfang der Dienstanforderung keinem anderen System zugeordnet, so wird die Kontrolle an die anfordernde Applikation übergeben. Alle bis zur Auflösung der Zuordnung eintreffenden Anforderungen bezüglich der Zuteilung der Kontrolle über diese VMMD-Instanz werden zurückgewiesen. Neben der expliziten Auflösung der Zuordnung durch Nutzung des ACCESS-CONTROL Dienstes wird diese auch selbständig durch den Server vorgenommen, wenn die Kommunikationsbeziehung abgebaut wird oder die Client-Applikation sich mittels des Dienstes UNSUBSCRIBE-CMD als Empfänger dieses Datenstromes abmeldet.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Mode	M	M(=)		
Result (+)			**S**	**S(=)**
Result (-)			**S**	**S(=)**
Error-specification			M	M(=)

Tabelle 25: Parameter der Dienstelemente des Dienstes ACCESS-CONTROL

Der dritte für VMMDs neu definierte Dienst (CODER-CONTROL) dient zur Übertragung von speziellen Kommandos zum Koder der Daten. Die Vielfalt existierender Kodierungsverfahren mit zusätzlich oftmals implementierungsspezifischen Steuerungsanweisungen, z.B. für besondere Kodierungs-Chipsätze, verlangt nach einer flexiblen Gestaltung des Nutzdatenteils dieses Dienstes. Das Steuerungskommando wird deshalb als Folge von Oktetten modelliert, so daß die Übertragung beliebiger Zeichensequenzen möglich ist. Am VMMD wird der Dienst durch Übergabe der Anweisung an die Kodierungseinheit ausgeführt und es wird entsprechend dem Erfolg der Dienstausführung eine positive bzw. negative Quittierung an den Initiator des Dienstes gesendet.

Aus dem VMMD werden durch Spezialisierung die Objekttypen des Multimedia-Gerätes für beliebige kontinuierliche Medien (CMD, Continous Media Device), für Systeme zur Aufnahme beliebiger visueller Informationen (CD, Camera Device) und für Standbilder (SID, Still Image Device) direkt abgeleitet. Objekte zur Repräsentation von Systemen zur Aufnahme auditiver Informationen und von Bewegtbildern werden mittels weiterer Spezialisierung definiert.

Beschreibung des Objekttyps „Camera Device"

Der Objekttyp Camera Device (CD) dient der Modellierung gemeinsamer Attribute von optischen Aufnahmesystemen, insbesondere Stand- und Bewegtbildkameras. Eine Instantiierung von Objekten dieses Objekttyps ist nicht direkt vorgesehen. Es sind die nachfolgend aufgeführten Attribute festgelegt:

Objekttyp = Camera-Device
Attribute : Zoom-range
Attribute : Focus-modes-supported [Auto, Manual, Both]
Attribute : Available-Shutter-speeds
Attribute : Current extended settings
VMMD :: Service : Set

Die zusätzlichen Attribute modellieren den Bereich der Brennweite des Objektivs der Kamera als Interval („Zoom-range"), die unterstützten Modi für den Fokus der Kamera („Focus-modes-supported") sowie die verfügbaren Verschlußzeiten („Available-Shutter-Speeds"). Die geerbte Ope-

ration „Set" (vgl. Tabelle 26) erlaubt neben der Vorgabe einer neuen Position („Position") und des Geschwindigkeitsprofiles, das zum Verfahren genutzt werden soll, zusätzlich das Setzen der durch das CD eingeführten Attribute. Deren aktuelle Werte werden in den Parametern „Current settings" und „Current extended settings" gehalten und sind den Clients über den Dienst `GET-ATTRIBUTES` zugänglich.

Beschreibung des SID

Für den Objekttyp des SID sind gegenüber dem CD keine weiteren Attribute notwendig. Es wird eine zusätzliche Operationen zum Anstoß der Aufzeichnung eines Bildes eingeführt:

Service : Take-Image

Die Ausführung des Dienstes `TAKE-IMAGE` resultiert in der Erstellung eines Domains mit der in der Dienstbestätigung angegebenen Bezeichnung. Die Dienstantwort wird nach der Erzeugung des Bildes und dessen Kodierung gesendet. Die Parameter der Dienstelemente sind in Tabelle 27 zusammengestellt. Durch Wahl eines Wertes für den Parameter „Auto-upload" kann der Dienstbenutzer bestimmen, ob das erzeugte Domain automatisch zu ihm geladen werden soll oder ob er

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Focus-mode	C	C(=)		
Shutter-speed	C	C(=)		
Position	M	M(=)		
Speed-profile	M	M(=)		
Result (+)			**S**	**S(=)**
Result (-)			**S**	**S(=)**
Error-specification			M	M(=)

Tabelle 26: Parameter der Dienstelemente des Dienstes `SET`

selbst die Übertragung explizit anfordern möchte. Diese Wahlmöglichkeit erlaubt es dem Dienstbenutzer auch, eventuell durch Programminstanzen erbrachte Vorverarbeitungen der Informationen, z.B. die Kompensation von Bildrauschen durch die Aufnahme und Verarbeitung mehrerer Bilder vom gleichen Objekt, vornehmen zu lassen.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Coding-technique	M	M(=)		
Coding extension	M	M(=)		
Auto-upload	M	M(=)		
Delete after	M	M(=)		
Result(+)			**S**	**S(=)**
Domain-ID			M	M(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 27: Parameter der Dienstelemente des Dienstes `TAKE-IMAGE`

Das Löschen eines erzeugten Bildes kann entweder automatisch nach dem erfolgreichen Heraufladen des erzeugten Domains zum Dienstbenutzer oder erst auf dessen explizite Anforderung hin erfolgen. Die Wahl zwischen den beiden Verfahren wird durch Setzen des entsprechenden Wertes für den Parameter „Delete after" erreicht.

Beschreibung des CMD

Der Objekttyp des CMD ist ein Untertyp des VMMD und besitzt selbst wiederum Untertypen zur Modellierung von Geräten zur Aufnahme von visuellen bzw. auditiven Informationen. Gegenüber dem VMMD verfügt es über ein neues Attribut und drei zusätzliche Operationen:

Attribute : List of Subscribers
Service : Subscribe CMD
Service : Unsubscribe CMD
Service : Re-negotiate

CMDs produzieren Datenströme, die im Regelfall isochron über das Netzwerk übertragen werden müssen. Diese Übertragung erfolgt in OSIRIS im Multicast unter Verwendung der Mechanismen des periodischen Datenaustausches. Die Kontrolle der Geräte erfolgt durch Dienste, die mittels der Mechanismen der Nachrichtenübertragung zwischen Stationen über Applikationsbeziehungen bzw. im Broadcast übertragen werden. Jede Station, die den Start einer Übertragung des Datenstromes mittels des Dienstes `SUBSCRIBE-CMD` anfordert, wird in die Liste der aktuellen Empfänger („List of Subscribers") übernommen. Diese beinhaltet Informationen über die unterstützten Kodierungsverfahren sowie die dazugehörigen Parameter. Als Zuordnung zu den Anforderungen genügt dabei eine interne Numerierung („Subscriber-ID"). Bild 76 stellt beispielhaft einen Auszug aus einer derartigen Liste dar, in der zwei Stationen als Empfänger eines Datenstromes geführt sind. Die erste aufgeführte Station unterstützt dabei die Kodierung von Bewegtbildern

Subscriber-ID	Coding-technique	Coding-parameter-range
1	H.261	$p \in [1, 5]$
2	H.261	$p \in [1, 3]$
.		
.		
.		

Bild 76: Auszug aus der Liste der Empfänger eines Multimedia-Datenstromes

nach dem H.261 Standard mit $p \in [1, 5]$, die zweite nur den Bereich $p \in [1, 3]$ (vgl. Kap. 2.6.1). Die Parameter der Dienstelemente des Dienstes `SUBSCRIBE-CMD` sind in Tabelle 28 zusammengefaßt.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Coding-technique	M	M(=)		
Desired Quality	M	M(=)		
Tolerable Range	M	M(=)		
Result(+)			**S**	**S(=)**
Stream-ID			M	M(=)
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 28: Parameter der Dienstelemente des Dienstes `SUBSCRIBE-CMD`

Bei der Bearbeitung der Dienstankündigung sind seitens des Servers folgende Fälle zu unterscheiden:

- Es liegt noch keine Anforderung vor. In diesem Fall wird die benötigte Bandbreite angefordert und entsprechend dem Erfolg dieses Vorganges eine positive oder negative Dienstbestätigung gesendet. Konnte die benötigte Bandbreite für die isochrone

Datenübertragung angefordert werden, so beginnt der Server mit der Erzeugung der Daten und stellt sie dem Kommunikationssystem zur Übertragung bereit. Dazu steht der Dienst `MM-TRANSMIT` zur Verfügung.

- Die angeforderte Kodierungsqualität entspricht derjenigen, die für die bereits vorhandenen Anforderungen aktuell benutzt wird. In diesem Fall wird eine positive Dienstbestätigung geschickt, in der die Identifikation des Multimedia-Datenstromes enthalten ist. Zudem wird die Liste der Empfänger entsprechend angepaßt.
- Die angeforderte Kodierungsqualität unterscheidet sich von der, die aktuell verwendet wird. In diesem Fall wird überprüft, ob die angeforderte Kodierungsqualität in der Schnittmenge der bereits berücksichtigten Toleranzbereiche liegt. Ist dies der Fall, so wird über Management-Dienste die benötigte Bandbreite angefordert (vgl. Kap. 7.5.3). Kann diese zur Verfügung gestellt werden, so werden eine positive Dienstbestätigung und eine Benachrichtigung (`RE-NEGOTIATE`) über die Änderung der Kodierungsgüte an den Requester bzw. an alle angeschlossenen Stationen gesendet. Liegt die angeforderte Kodierungsgüte nicht in der Schnittmenge bereits berücksichtigter Anforderungen oder kann die Übertragungsbandbreite nicht bereitgestellt werden, so wird eine negative Quittung an den Requester gesendet.

Durch Aufruf des `UNSUBSCRIBE-CMD` Dienstes (vgl. Tab. 29) kann ein Client dem Server mitteilen, daß er nicht weiter als Senke des spezifizierten Datenstrom auftritt. Er übergibt zudem noch

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Object-ID	M	M(=)		
Stream-ID	M	M(=)		
Tolerable Range	M	M(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 29: Parameter der Dienstelemente des Dienstes `UNSUBSCRIBE-CMD`

den von ihm bei der Initiierung des Dienstes angeforderten Parameterbereich. Mit Hilfe dieser Informationen wird die Liste der Empfänger aktualisiert. Ist keine Anforderung mehr vorhanden, so wird die Erzeugung von Daten für den Datenstrom gestoppt und die Übertragungsbandbreite freigegeben. Eine negative Dienstbestätigung wird nur dann gesendet, wenn der spezifizierte Datenstrom nicht existiert.

Wird auf Anforderung eines neuen Empfängers hin eine Änderung der Parameter der Kodierung vorgenommen, z.B. eine Reduzierung des Parameters „p" der H.261 Bewegtbildkodierung von „3" auf „2", so sendet der Server selbsttätig aperiodisch mit hoher Priorität eine Benachrichtigung an die Clients. Dazu steht der unbestätigte Dienst `RE-NEGOTIATE` zur Verfügung, dessen Parameter in Tabelle 30 aufgeführt sind.

Neben der Bezeichnung des Datenstromes wird der neue Wert des Parameters übertragen, der die Kodierungsgüte beschreibt. Dieser ist abhängig von dem jeweils verwendeten Kodierungsverfahren im entsprechenden Kontext zu interpretieren. Die Rekonfigurierung des Kodierungsalgorithmus wird seitens des Erzeugers des Datenstromes erst nach erfolgter Absendung der Dienstanforderung vorgenommen.

Parameter Name	ind
Argument	**M**
Stream-ID	M
New Quality	M

Tabelle 30: Parameter der Dienstelemente des Dienstes `RE-NEGOTIATE`

Zur Übergabe von gemäß den Anforderungen kodierten Daten aus multimedialen Datenströmen wird der Dienst `MM-TRANSMIT` genutzt. Es handelt sich um einen unbestätigten Dienst, dessen Dienstparameter in Tabelle 31 aufgeführt sind.

Die Daten werden im lokalen Speicher der Quellstation unter Verwaltung von DRSLP bis zur Übertragung zwischengespeichert. Die Sequenznummer kann von der konsumierenden Applikation zur Erkennung des Verlustes oder der mehrfachen Übertragung einer Dateneinheit genutzt werden. Sie wird modulo 2^8 inkrementiert.

Parameter Name	req	ind
Argument	**M**	**M(=)**
Stream-ID	M	M(=)
Length	M	M(=)
Data	M	M(=)
Sequence Number	M	M(=)

Tabelle 31: Parameter der Dienstelemente des Dienstes `MM-TRANSMIT`

Systeme zur Aufnahme auditiver Daten lassen sich durch das CMD bereits vollständig modellieren und erben dessen Attribute und Operationen. Systeme zur Aufnahme von Bewegtbildern erben zusätzlich die Attribute und Operationen der Objektklasse CD.

6.3 Kodierung der zu übertragenden Daten

Zur Gewährleistung einer rechner- und betriebssystemunabhängigen Repräsentation von Daten findet deren Übertragung in einer definierten Transfersyntax [8] statt. Die Verwendung nur eines Darstellungskontextes, der die Kombination einer abstrakten Syntax und einer Transfersyntax bezeichnet [8], macht ein Protokoll zur Aushandlung eines Darstellungskontextes überflüssig.

Es ist zwischen der Kodierung von Variablen, Protokolldateneinheiten von RTSIMS und des Management-Protokolls (vgl. Kap. 7.1) und Daten aus Multimedia-Datenströmen zu unterscheiden. Da die Spezifikation des Typs und der Speicherungsstruktur jeder Variablen, die in einer Station produziert oder konsumiert wird, dort vorliegt, entfällt die Notwendigkeit von dessen Kodierung. Für die Kodierung einfacher Variablen werden die in den Kodierungsregeln für PROFIBUS festgelegten Definitionen übernommen. Die Werte der Komponenten strukturierter Variablen werden direkt aufeinanderfolgend übertragen, wobei jeder einzelner Wert gemäß der Konventionen für einfache Variable dargestellt wird. Die zu einer Variablen gehörigen Informationen, die zwischen den Kommunikationspartnern über die verteilte Datenbasis ausgetauscht werden, umfassen neben dem Prozeßwert weitere Informationen. Zusammen bilden sie einen Verbund, der in der Reihenfolge der Definition der einzelnen Komponenten gespeichert und als Block übertragen wird.

Die Spezifikation der Protokolldateneinheiten, die zwischen Instanzen der Protokolle der Anwendungsschicht ausgetauscht werden, in ASN.1 (vgl. Anhang C.2 bzw. D) erlaubt die Verwendung von bereits für diese abstrakte Syntax definierten Kodierungsregeln. Aus Gründen von deren mangelnder Performance [147] wird dabei nicht auf die *Basic Encoding Rules for ASN.1* [148] zurück-

gegriffen, sondern es werden die diesbezüglichen Festlegungen für PROFIBUS, die *Fieldbus Encoding Rules*, die bereits für die Verwendung im Feldkommunikationsbereich optimiert wurden [111], übernommen. Für die Darstellung der Zeitinformationen gelten dabei folgende Festlegungen:

- Eine *Uhrzeit*, z.B. der Erzeugungszeitpunkt eines Wertes, wird in vier aufeinanderfolgenden Oktetten verschlüsselt. Das höchstwertige Byte wird dem ersten Oktett zugewiesen. Entsprechend ihrer Wertigkeit folgen die anderen Bytes der Uhrzeit. In jedem Byte ist das achte Bit als höchstwertiges Bit festgelegt.
- Eine *Gültigkeitszeitdauer* wird grundsätzlich mit den Werten der Attribute „Produced At" und „Remote Validity", sowie der Synchronisationsgüte in vier aufeinanderfolgenden Oktetten übertragen. Die Belegung der Bitpositionen ist in Bild 77 veranschaulicht.
- Als einzelnes Datum wird die *Synchronisationsgüte* in Form einer *Unsigned8* Variable kodiert.

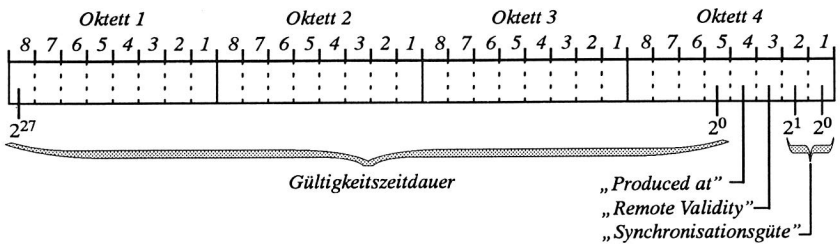


Bild 77: Kodierung der Gültigkeitszeitdauer zusammen mit weiteren Attributen

Bezüglich der Daten, die von durch VMMDs modellierten Aufnahmesystemen erzeugt werden, ist zu beachten, daß hier eine Festlegung des Formates bereits auf Ebene der Anwendung erfolgt. So wird z.B. durch einen Koder für H.261 auf Ebene der Anwendung ein Datenstrom erzeugt, der durch das Kommunikationssystem nur noch transportiert, aber nicht mehr inhaltlich modifiziert werden muß. Dies kann durch eine Modellierung der erzeugten Nutzdaten als Bitfolge erreicht werden. Für die Übertragung von Standbildern werden Domains verwendet. Diese besitzen eine beliebige Länge und werden durch das Kommunikationssystem in genau der Form übertragen, in der sie vorliegen. Eine weitere Kodierung durch das Kommunikationssystem ist nicht notwendig.

6.4 Mindestumfang einer Implementierung von RTSIMS

Das RTSIMS-Protokoll bietet – wie ausgeführt und durch Bild 78 veranschaulicht – eine Vielzahl von Diensten an. Es ist dabei nicht zwingend erforderlich, daß in jeder Implementierung alle Dienste unterstützt werden. Für einfache Stationen, z.B. Sensoren oder Aktoren mit dezentraler Signalverarbeitung, erweist sich die Einschränkung auf Dienste zum Austausch von Variablen in der Regel als ausreichend [149]. Unter Verzicht auf den Modus „update" beim Lesedienst kann dies durch alleinige Implementierung der Dienste `READ` bzw. `WRITE` erreicht werden. Entsprechende Vereinfachungen ergeben sich in diesem Fall auch für das LLI, in dem dann keine Verbindungsverwaltung benötigt wird.

Ist der Austausch von Nachrichten notwendig, so müssen mindestens die Dienste `INITIATE` und `ABORT` implementiert sein. Für den Server ist beim Dienst `INITIATE` dabei die Rolle des Re-

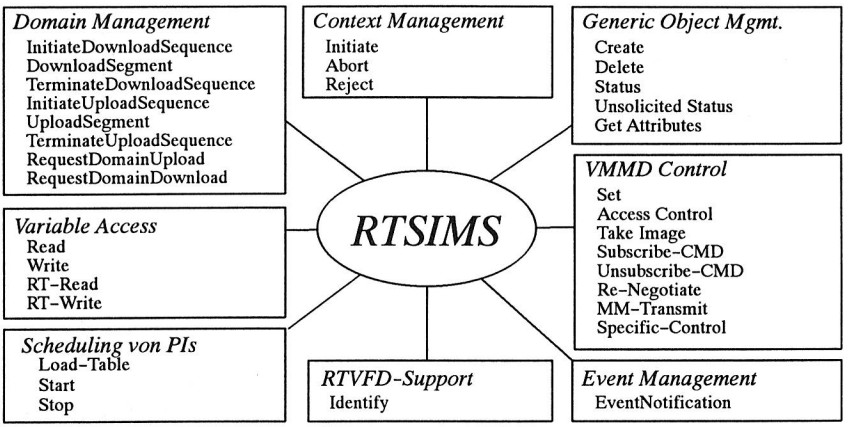


Bild 78: Dienste und logische Dienstgruppen in RTSIMS

sponders vorgeschrieben. Der Dienst ABORT muß sowohl als Requester als auch als Responder erbracht werden. Alle weiteren Dienste sind optional. Dabei ist die Berücksichtigung von zusammengehörigen Diensten, z.B. beim Laden von Domains, zu beachten.

6.5 Zusammenfassung der wesentlichen Eigenschaften von RTSIMS

RTSIMS vereinigt als erstes Protokoll seiner Art die Funktionalität zur Steuerung von Fertigungsgeräten und Multimedia-Aufnahmegeräten. Dabei bietet aber auch schon allein der Teil des Protokolls, mit dem Fertigungsgeräte gesteuert werden, gegenüber den vorhandenen Protokollen, z.B. MPS von FIP und FMS von PROFIBUS, eine

- durch die integrierte Behandlung von Zeit,
- die Bevorratung von Mechanismen zum lokalen, aber dennoch global synchronisierten, zeit- und ereignisgesteuerten Aufruf von Applikationen
- und ein komfortables Ereignisverwaltungssystem

deutlich erweiterte Funktionalität im Sinne der Erfüllung der aufgestellten Anforderungen. Die Steuerung von Aufnahmegeräten für multimediale Informationen ist in einem für den prozeßnahen Bereich der rechnerintegrierten Fertigung adäquaten Maß integriert. Auf Basis der definierten abstrakten Geräte für die Aufnahme und übertragung von kontinuierlichen Medien können auch Applikationen wie *Bedienerkommunikation im Gegensprechen* oder *Videokonferenzen* realisiert werden. Die Flexibilität des gewählten Ansatzes gestattet es beispielsweise auch, durch Modellierung von z.B. CD-ROM-Laufwerken als Datenquelle, Bewegtbildsequenzen oder Standbilder als ergänzende Information bei der Wartung von technischen Systemen, über das Netzwerk abzurufen.

Der bei der Konzeption von RTSIMS verfolgte objektorientierte Ansatz erlaubt, im Vergleich zu funktional hochwertigen Protokollen der Anwendungsschicht anderer Feldkommunikationssysteme, eine Reduktion der Anzahl verschiedener Dienste. An deren Aufrufschnittstelle stehen Parameter zur Steuerung der Dienstgüte in Hinblick auf die zu verwendenden Mechanismen des Datenaustausches und der Priorität der Dienstausführung zur Verfügung. Die Zeitbehaftung von Daten wirkt sich auf Variable und auf Ereignisse aus.

7 Struktur und Funktion des Systemmanagements

Zur Gewährleistung der Operabilität des Systems ist für OSIRIS neben den Protokollen zum Austausch von Daten zwischen benutzerdefinierten Anwendungen ein Protokoll für das Systemmanagement festgelegt, das im folgenden näher beschrieben wird.

7.1 Überblick

Für OSIRIS ist Funktionalität des Systemmanagements [150] festgelegt. Ein Schichtmanagement (Layer Management [8]) wird nicht benötigt. Die Managementfunktionalität wird im wesentlichen durch den *Manager* und die *Agenten* erbracht. Dabei existieren in jedem OSIRIS-System genau ein aktiver Manager, der der Buskopfstation zugeordnet ist, und in jeder anderen Station ein Agent sowie evtl. redundante Manager. Die Agenten führen einerseits auf Anforderung des Managers Operationen auf den verwalteten Objekten (*managed objects*) der jeweiligen Station aus und sind anderseits in der Lage, Mitteilungen und - in begrenztem Maße - auch Anforderungen, an den Manager zu senden. Sowohl der Manager als auch die Agenten sind Anwendungen und nutzen die Dienste, die ihnen vom Management-Protokoll (OMP, OSIRIS Management Protocol) zur Verfügung gestellt werden. Dieses ist Bestandteil der Anwendungsschicht. Die für dieses Protokoll definierten Dienste besitzen das Präfix „SM-" und werden grundsätzlich auf den Dienst DTU des LLI-Protokolls am LLI-SAP „2" abgebildet.

Der Austausch von Managementinformationen wird mittels definierter Management-Protokolldateneinheiten (MAPDUs) über die für das Systemmanagement reservierte Kommunikationsbeziehung mit der Kommunikationsreferenz „0" sowie mittels des periodisch ausgetauschten Managementrahmens vorgenommen (vgl. Bild 79). Die Kodierung der Protokolldateneinheiten geschieht wie in Kapitel 6.3 für die Protokolldateneinheiten von RTSIMS dargestellt. Die Spezifikation der unterschiedlichen MAPDUs in ASN.1 ist in Anhang D dargestellt.

Auch bei bestätigten Diensten sendet der Manager die Protokolldateneinheiten grundsätzlich im Broadcast an alle Stationen. Erst der Agent nimmt eine Auswertung der in der Protokolldateneinheit enthaltenen Zieladresse vor. Agenten senden ihre Anforderungen bzw. Bestätigungen grundsätzlich an die Stationsadresse „128", damit sie an all die Stationen übertragen wird, die potentiell

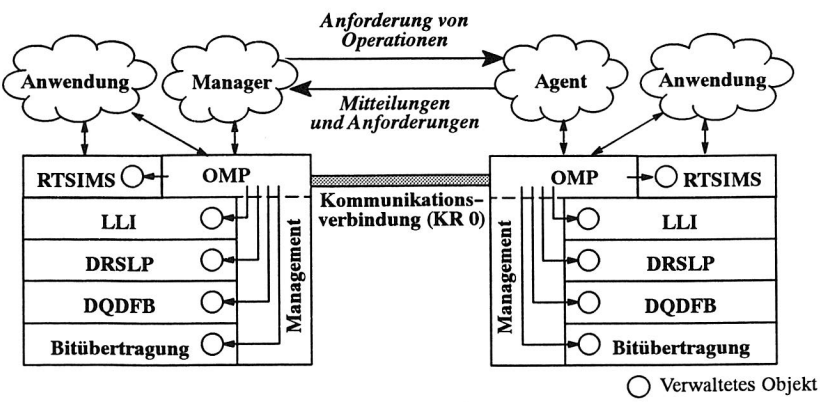


Bild 79: Struktur des Managements in OSIRIS

als BKS auftretenden können. Auf eine Dienstanforderung hin führen grundsätzlich nur der adressierte Agent bzw. der aktive Manager Funktionen aus.

Aus den funktionalen Bereichen des Managements, die in den Normen zum Netzwerkmanagement [151] spezifiziert sind, beinhaltet OSIRIS Funktionalität zur Erbringung von *Anwendungsmanagement, Fehlermanagement, Konfigurationsmanagement* und *Leistungsmanagement*. Darüber hinaus steht ein *Uhrzeitmanagement* zur Verfügung (vgl. Bild 80).

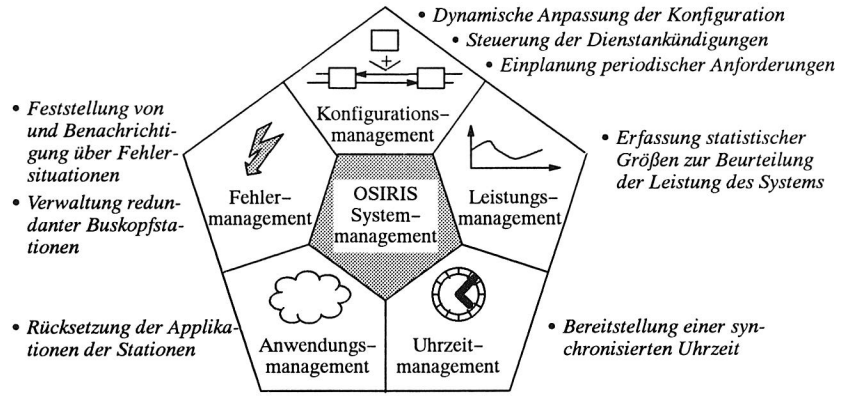


Bild 80: Übersicht über die Funktionskomponenten des Systemmanagements in OSIRIS

Zu den verwalteten Objekten einer Station gehören die Instanzen der Kommunikationsprotokolle und die zur Erbringung des Fehler-, Konfigurations- und Leistungsmanagements notwendigen Zähler und Tabellen, die nachfolgend noch näher spezifiziert werden. Diese Informationen werden in der MIB einer jeden Station jeweils lokal gehalten, in der auch allgemeine Informationen, wie z.B. die Latenzzeit der Station, die eigene Stationsadresse und die Adressen der Gruppen, denen die Station angehört, enthalten sind.

7.2 Generische Objekte und Dienste

Viele der Objektinstanzen lassen sich unabhängig von ihrer Ausprägung für Zwecke des Managements einheitlich beschreiben. Dazu gehören die Instanzen der unterschiedlichen Kommunikationsprotokolle und die verschiedenen Arten von Zählern.

Instanzen von Kommunikationsprotokollen und Anwendungen lassen sich über den Dienst `SM-RESET` in ihren Initialzustand zurücksetzen. Zu den Parametern dieses nur durch den Manager zu initiierenden bestätigten Dienstes gehören die Bezeichnung der Protokollinstanz bzw. Anwendung und die Angabe der Priorität (vgl. Tab. 32). Die Bezeichnung der in einer Station definierten Anwendungen wird in dieser in der MIB geführt. Die Adresse der anzusprechenden Station wird an der Diensteschnittstelle ebenfalls übergeben. Dabei besteht auch die Möglichkeit, alle Protokollinstanzen und die Anwendungen einer Station mittels eines einzigen Dienstaufrufes zurückzusetzen.

Die Bestätigung wird vor der Ausführung der zum Zurücksetzen notwendigen lokalen Funktionen gesendet. Die Protokollinstanzen selbst bieten an ihrer Schnittstelle zum Management entspre-

chende Dienste an. Das Rücksetzen der laufenden Applikationen, z.B. der RTSIMS Server-Applikation wird über die Schnittstelle zum Betriebssystem der Station vorgenommen.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Instance-ID	M	M(=)		
Transfer Mode	M	M(=)		
Priority	M	M(=)		
Station-Address	M	M(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 32: Parameter der Dienstelemente des Dienstes SM-RESET

Bei Zählern wird grundsätzlich zwischen solchen unterschieden, die nur modulo eines maximalen Wertes inkrementiert, in- und dekrementiert oder zusätzlich zum Inkrementieren beliebig gesetzt werden können. Die beiden erstgenannten Arten von Zählern läßt sich nur auslesen und zurücksetzen, letztgenannte lassen sich auch beschreiben. Alle Zähler haben initial den Wert „0" und können eindeutig über einen symbolischen Bezeichner angesprochen werden. Sie werden teilweise durch die Instanzen der Kommunikationsprotokolle geführt. Diese bieten an ihrer Schnittstelle zum Management Funktionen zum Zugriff auf die Zähler an. Das Auslesen eines Zählerstandes wird mittels des bestätigten Dienstes `SM-CREAD` vorgenommen. Die Dienstbestätigung enthält im Falle der erfolgreichen Ausführung den Zählerstand, anderenfalls eine Fehlerkennung. Zähler können mittels des bestätigten Dienstes `SM-CWRITE` gesetzt werden, sofern sie dafür vorgesehen sind. Der bestätigte Dienst `SM-CRESET` dient dem Zurücksetzen von Zählern.

7.3 Anwendungsmanagement

Das Anwendungsmanagement wird durch den Dienst `SM-RESET` gebildet. In dem OSIRIS-System erweist sich dieses als ausreichend, da die hier relevanten RTSIMS-Server-Applikationen durch das Laden von Program Invocations und Schedulingtabellen selbst über weitgehende Steuerungsmöglichkeiten bzgl. der Ausführung von Teilen der Anwendung verfügen.

7.4 Fehlermanagement

Das Fehlermanagement beinhaltet zwei Bestandteile: Die Übertragung allgemeiner Fehlermeldungen und die dezentrale Verwaltung redundanter Buskopfstationen, die nach dem Ausfall der aktiven BKS selbständig deren Rolle übernehmen können.

7.4.1 Übertragung von Fehlermeldungen

Informationen über von den Protokollinstanzen als verwaltete Objekte festgestellte Fehlersituationen können mittels des unbestätigten, hochprior übertragenen Mitteilungsdienstes `SM-FAULT-NOTIFICATION` an den Manager übermittelt werden. Die Parameter der Dienstelemente sind in Tabelle 33 dargestellt und umfassen eine Spezifikation des aufgetretenen Fehlers, die Angabe der Zeit, zu der das Ereignis festgestellt wurde und optional zusätzliche Informationen.

Zu den durch die Bitübertragungsschicht feststellbaren Fehlersituationen gehören beispielsweise der Ausfall einer oder beider Duplexverbindungen zu den Nachbarstationen. Durch DQDFB können insbesondere die Überläufe der Warteschlangen für vor der Übertragung zwischenzuspei-

chernde Protokolldateneinheiten, der Überlauf der Request-Counter und das Ausbleiben von Übertragungsrahmen über einen längeren Zeitraum festgestellt werden. DRSLP meldet z.B. dann einen Fehler, wenn über die verteilte Datenbasis auf Variable zugegriffen wird, die nicht definiert sind.

Parameter Name	req	ind
Argument	**M**	**M(=)**
Fault-ID	M	M(=)
OccurrenceTime	M	M(=)
Synchronisation Quality	M	M(=)
Additional Detail	C	C(=)

Tabelle 33: Parameter der Dienstelemente des Dienstes `SM-FAULT-NOTIFICATION`

Zu den durch das OMP an sich feststellbaren Fehlern gehören insbesondere Verletzungen der Konsistenz der Konfiguration, z.B. die Verwendung der eigenen Stationsadresse durch eine andere Station oder der Empfang einer Nachricht von einer neuen Vorgängerstation.

7.4.2 Redundante Buskopfstationen

Die Buskopfstation spielt durch ihre Funktion als Rahmenquelle und -senke sowie durch die Verwaltung der Konfiguration eine zentrale Rolle in einem OSIRIS-System. Ihr Ausfall bedeutet, sofern nicht entsprechende Vorkehrungen getroffen werden, den Ausfall des gesamten Systems.

In OSIRIS ist deshalb die Möglichkeit vorgesehen, die Funktion der BKS und die des dort laufenden Manager-Prozesses redundant in mehreren Station vorzusehen, von denen jedoch grundsätzlich nur eine aktiv ist. Alle redundanten BKS führen zur Laufzeit des Systems jedoch alle Informationen, z.B. die Pollingtabelle, genauso wie die aktive Station, senden dabei aber weder Dienstbestätigungen noch stellen sie selbst Dienstanforderungen in der Rolle des Managers. Dies wird dadurch ermöglicht, daß bei Anfragen an den Manager-Prozeß der Buskopfstation grundsätzlich die hierfür reservierte Gruppenadresse („128") verwendet wird, so daß diese Information auch in allen als redundante BKS fungierenden Stationen vorliegt und dort lokal verarbeitet werden kann.

Der Ausfall der BKS wird durch Ausbleiben des Eintreffens von Übertragungsrahmen über den Bus von jeder Station festgestellt. Die Wahl einer neuen BKS geschieht dezentral. Um zu vermeiden, daß zwei Stationen unabhängig voneinander quasi-gleichzeitig versuchen, die Rolle der aktiven BKS zu übernehmen, wird der Start der BKS-Funktionalität in einer Station erst dann vorgenommen, wenn innerhalb einer festgelegten Zeitspanne kein Übertragungsrahmen einer anderen Station empfangen wurde. Diese Verzögerungszeit t_{VBKS} berechnet sich in Abhängigkeit von der Stationsnummer k, der Anzahl der Stationen n im System und der Latenzzeit wie folgt:

$$t_{VBKS}(n, k, t_{Lat}) = \frac{1}{2} \times (k + 1) \times n \times t_{Lat} \qquad (12)$$

Diese Wahl von t_{VBKS} sorgt dafür, daß Stationen mit niedriger Stationsnummer Vorzug vor denen mit hoher Stationsnummer haben, wobei vor dem Start der BKS-Funktionalität mindestens solange gewartet wird, bis ein von einer beliebigen anderen Station im Netz gesendeter Übertragungsrahmen eingetroffen wäre.

7.5 Konfigurationsmanagement

Das Konfigurationsmanagement hat in OSIRIS die Aufgabe, alle zur Wahrung der Konsistenz der Konfiguration notwendigen Maßnahmen durchzuführen sowie eine Schnittstelle zum Setzen der

Parameter des Filters für die im Zusammenhang mit der Verwaltung der verteilten Datenbasis und Multimedia-Datenströmen eintreffenden Dienstankündigungen zur Verfügung zu stellen. Als dritte wesentliche Komponete des Konfigurationsmanagements ist die dynamische Verarbeitung von Anforderungen für Kommunikationsbandbreite zur Übertragung periodischer Informationseinheiten zu nennen.

7.5.1 Wahrung der Konsistenz der Konfiguration

In jeder OSIRIS-Station liegt eine Erreichbarkeitstabelle vor, in der aufgeführt ist, über welchen der gerichteten Busse eine Protokolldateneinheit gesendet werden muß, damit sie eine bestimmte Zielstation erreicht. Die Erreichbarkeitstabellen werden dynamisch bei der Inbetriebnahme des Systems aufgebaut und während des Betriebes laufend an Konfigurationsänderungen angepaßt.

Zum Aufbau der Erreichbarkeitstabellen bei der Inbetriebnahme des Systems sendet die BKS auf beiden Bussen solange „Power-up"-Übertragungsrahmen (vgl. Anhang D), bis alle Stationen im System aufgenommen sind. Dazu schreibt jede Station auf beiden Bussen in den ersten freien Management-Power-up-Rahmen ihre Adresse. Aus empfangenen Rahmen, die bereits die Adresse einer Station enthalten, die auf dem Bus näher an der BKS positioniert ist, entnehmen die Stationen Informationen über die Anordnung der auf den Bussen zwischen ihnen und der BKS liegenden Stationen. Die BKS selbst baut ebenfalls eine Erreichbarkeitstabelle auf. Sie stoppt die Erzeugung von „Power-up"-Übertragungsrahmen, wenn sie für beide Busse über zueinander konsistente Informationen verfügt und auf beiden Bussen mindestens ein derartiger Rahmen ohne Stationseintrag wieder bei ihr eingetroffen ist. Bild 81 stellt die Inhalte der Erreichbarkeitstabellen an einem Beispiel dar. Grundsätzlich kann die BKS von allen Stationen über beide Busse erreicht werden; sie kann ihrerseits alle Stationen über beide Busse erreichen.

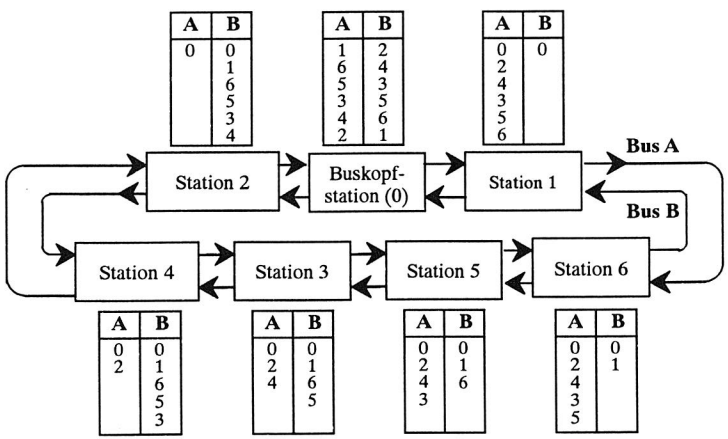


Bild 81: Beispielkonfiguration mit Erreichbarkeitstabellen

Zu Wahrung der Konsistenz im laufenden Betrieb sendet die BKS periodische Management-Rahmen (vgl. Anhang D) auf beiden Bussen, in denen Stationen ihre Aufnahme in das System oder ihr geplantes Ausscheiden ankündigen können. Im Normalbetrieb, d.h. wenn weder Ausgliederungs- noch Aufnahmeanforderungen vorliegen, lesen die Stationen diese Rahmen aus, überprüfen die Konsistenz ihrer Informationen und schreiben die Stände ihrer Request-Zähler für den

betroffenen Bus sowie ihre Adresse vor der Weitergabe an die nächste Station in den Management-Rahmen.

Zur Ein- und zur Ausgliederung einer Station stellt das OMP dem Agenten die Dienste SM-INCLUDE-STATION und SM-EXCLUDE-STATION zur Verfügung. Beide besitzen als einzigen Parameter die Adresse der bezogenen Station. Die eigentliche Abwicklung der Ein- bzw. Ausgliederung geschieht mittels der in den periodischen Management-Rahmen ausgetauschten Informationen.

Bei zur Laufzeit des Systems vorliegendem Eingliederungswunsch wird durch die Station in einem, noch nicht für andere Rekonfigurationswünsche benutzten, periodischen Management-Rahmen das Rekonfigurationsbit auf „1" gesetzt. Zudem werden die eigene Adresse und die ihrer Vorgängerstation in die entsprechenden Felder eingetragen. Dieser Vorgang wird auf beiden Bussen durchgeführt. Alle anderen Stationen nutzen die Informationen zur Aktualisierung ihrer Erreichbarkeitstabellen und zur Aktualisierung der kumulierten Latenzzeit für die Uhrzeitsynchronisation, sofern sie auf Bus A weiter von der BKS entfernt sind, als die eingliederungswillige Station. Die Erreichbarkeitstabelle für die einzugliedernde Station wird von der BKS vorbereitet und mittels des bestätigten Dienstes SM-DOWNLOAD-TABLE des Systemmanagements an die Station heruntergeladen. Dieses Laden geschieht unter der Verwendung eines einzigen Übertragungsrahmens sowie der zugehörigen Bestätigung. Zur Wahrung der Konsistenz der Erreichbarkeitstabellen werden erst nach Abschluß des Ladens der Erreichbarkeitstabellen wieder Management-Rahmen gesendet, mit denen Stationen ihre Aufnahme oder ihre Ausgliederung beantragen können. Dadurch, daß in den periodischen Management-Rahmen die aktuellen Stände der Request-Zähler der jeweiligen Vorgängerstation für den betroffenen Bus eingetragen sind, kann eine eingliederungswillige Station auch ihre Request-Zähler korrekt setzen und nach ihrer vollständigen Eingliederung an dem Zugriffsverfahren auf QA-Rahmen teilnehmen.

Besteht bei einer Station der Wunsch, sich aus dem System auszugliedern, so setzt sie in einem Management-Rahmen, der noch nicht für andere Aus- oder Eingliederungen genutzt wird, das Ausgliederungsbit auf „1" und fügt ihre Adresse (EA, Exclude-Address) ein. Die anderen im System befindlichen Stationen streichen dann die Adresse aus den Erreichbarkeitstabellen und aktualisieren, sofern die auszugliedernde Station auf Bus A der BKS näher ist als sie selbst, die kumulierte Latenzzeit.

Es besteht zudem die Möglichkeit, daß Stationen im laufenden Betrieb unkontrolliert ausfallen und sich deshalb nicht selbst ordnungsgemäß aus dem System ausgliedern können. Ein derartiger Fall kann auf Basis der periodischen Übertragung der Management-Rahmen festgestellt werden. Hierzu überprüft eine Station, ob die eingetragene Adresse der Vorgängerstation gleich dem letzten Eintrag der Erreichbarkeitstabelle für den Bus in entgegengesetzter Richtung ist. Ist dies nicht der Fall, so kann sie anhand der in dieser Tabelle gespeicherten Informationen erkennen, welche Stationen ausgefallen sind. Sie übernimmt dann das Ausgliedern der Stationen. Dazu nutzt sie für alle zwischen ihr und ihrer neuen Vorgängerstation ausgefallenen Stationen jeweils einen periodischen Management-Rahmen. In diesem setzt sie das Ausgliederungsbit auf „1" und fügt die Adresse der ausgefallenen Station ein. Eine Korrektur der Latenzzeiten erfolgt damit ebenfalls automatisch.

Das Systemmanagement stellt ergänzend lokale Funktionalität zum Erfragen der Identifikation des Busses zur Verfügung, über den eine Protokolldateneinheit gesendet werden muß, damit diese

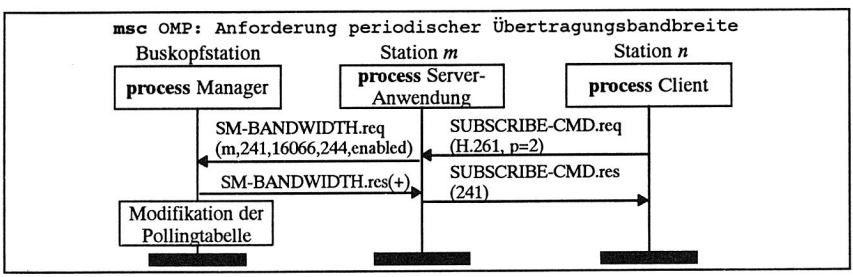
die Zielstation erreichen kann (SM-GET-BUS-ID). Des weiteren bietet sie einen lokalen Dienst zur Abfrage der Liste der aktuell in der Konfiguration befindlichen Stationen (SM-LIFELIST).

7.5.2 Filterung von Dienstankündigungen

Die Aktualisierung der lokalen Kopie von Variablen in der verteilten Datensbasis führt ebenso wie der Austausch von Multimedia-Datenströmen zu lokal generierten Dienstankündigungen, die für die aperiodisch zum Replikationsmechanismus ablaufenden Applikationen wichtig sein können. Um eine Beschränkung auf die für die jeweilige Applikation relevanten Dienstankündigungen zu ermöglichen, bevorratet das Systemmanagement einen Dienst, der auf die Filterfunktion des LLI wirkt. Dieser lokale Dienst (SM-INDICATION-CONTROL) erlaubt es einer lokalen Applikation, als Dienstbenutzer spezifische Dienstankündigungen für sich zu sperren oder freizugeben. Initial sind alle Dienstankündigungen gesperrt. Als Parameter wird die Kennung der Variablen bzw. des Multimedia-Datenstromes und die Angabe, ob es sich um eine Sperrung oder eine Freigabe handelt, an der OMP-Schnittstelle übergeben.

7.5.3 Dynamische Bandbreitenreservierung

Das Konfigurationsmanagement erlaubt es den angeschlossenen Stationen, dynamisch Anforderungen bezüglich Übertragungsbandbreite für periodische Datenübertragung zum Austausch von Variablen bzw. von zu Multimedia-Datenströmen gehörenden Informationseinheiten zu stellen. Diese werden vom Manager des OSIRIS-System entgegengenommen und bearbeitet. Der prinzipielle Ablauf einer derartigen Anforderung ist beispielhaft in Bild 82 dargestellt.



SUBSCRIBE-CMD.req(H.261, p=2):	*Anforderung der Übertragung eines H.261-kodierten Videodatenstromes mit einer Rate von 128 kbps.*
SM-BANDWIDTH.req	*(m,241,16066,244,enabled): Anforderung der periodischer Übertragungsbandbreite für Pufferplatz 241; 16066 Byte/s, Size = 244 Byte, sofortige Bereitstellung und Abarbeitung („enabled")*
SM-BANDWIDTH.res(+):	*Positive Dienstbestätigung*
SUBSCRIBE-CMD.req(241):	*Positive Dienstbestätigung Pufferplatzkennung*

Bild 82: Bandbreitenanforderung bei der Buskopfstation

In diesem Beispiel fordert eine Applikation, die im Anwendungsprotokoll und in der Anwendung als Server agiert, auf Anforderung eines Clients bezüglich der Übertragung von Videodaten die benötigte Bandbreite an. Für diesen Zweck steht der Dienst SM-BANDWIDTH zur Verfügung, dessen Parameter in Tabelle 34 dargestellt sind. Bei der Ausführung dieses Dienstes übernimmt die Applikation somit die Rolle des Requesters und der Manager die des Responders.

Der Dienst `SM-Bandwidth` erlaubt dabei durch Wahl des entsprechenden Wertes für den Parameter „Mode" sowohl die Anforderung als auch die Freigabe von periodischer Übertragungsbandbreite. Dabei kann unterschieden werden, ob bei möglicher Einplanung sofort mit der Abarbeitung der Einträge begonnen werden soll oder ob vorerst nur eine Reservierung der Bandbreite ohne deren weitere Berücksichtigung vorgenommen werden soll. In letzterem Fall werden die Einträge als „Disabled" gekennzeichnet (vgl. Kapitel 5.3.5). Mittels des Parameters „Buffer ID" wird die Pufferplatzidentifikation, bestehend aus Stationsadresse und Bezeichnung der Variablen bzw. des Multimedia-Datenstromes, angegeben. Die angeforderte Übertragungsbandbreite wird im Parameter „Rate" in der Einheit *Byte pro Sekunde*, die maximale Größe einer Übertragungseinheit im Parameter „Size" in *Byte* spezifiziert. Der maximale Wert für den Parameter „Size" ergibt sich aus der maximalen Länge der Nutzdaten einer IMPDU abzüglich der Länge der Verwaltungsinformationen und beträgt 244 Byte. Sollen Dateneinheiten aus Multimedia-Datenströmen in einem Segment übertragbar sein, so darf ihre Länge 27 Byte nicht überschreiten. Aus den Werten der Parameter „Size" und „Rate" kann der Manager die Anzahl von Rahmen berechnen, die er zur Befriedigung der Anforderungen senden muß.

Parameter Name	req	ind	res	cnf
Argument	**M**	**M(=)**		
Source Address	M	M(=)		
Buffer ID	M	M(=)		
Rate	M	M(=)		
Size	M	M(=)		
Mode	M	M(=)		
Transfer Mode	M	M(=)		
Priority	M	M(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**
Error Type			M	M(=)

Tabelle 34: Parameter der Dienstelemente des Dienstes `SM-BANDWIDTH`

Bei der Anforderung der Dienstgüte für den periodischen Austausch von Daten ist dabei durch die Anwendung ein Kompromiß aus guter Ausnutzung der Bandbreite und akzeptabler Verzögerung der Dateneinheiten in der Quell- und Zielstation einzugehen. Die wesentlichen Charakteristika, die Anzahl N_R benötigter PA-Übertragungsrahmen pro Sekunde und die durch die Größe der Übertragungseinheit bestimmte minimale Verzögerung D_{min} der Übertragungseinheit in der Quellstation, lassen sich bei gegebener Rate f_{DS} des zu übertragenden Datenstromes für OSIRIS wie folgt berechnen:

Minimale Paketierungsverzögerung in der Quellstation:

$$D_{min} = \frac{Size \times 8}{f_{DS}} \qquad (13)$$

Anzahl benötigter Übertragungseinheiten pro Sekunde

$$N_{ÜE} = \frac{f_{DS}}{(27 + \left\lceil \frac{Size-27}{31} \right\rceil \times 31) \times 8} \qquad (14)$$

In Bild 83 sind die Abhängigkeiten von D_{min} und N_R am Beispiel der Übertragung eines Videodatenstromes mit f_{DS} = 128 kbps dargestellt. Es zeigt sich, daß durch längere Übertragungseinheiten die Anzahl benötigter PA-Rahmen schrittweise abnimmt, die Verzögerung in der Quellstation

Anzahl benötigter PA-Rahmen für eine Multimedia-Dateneinheit

$$N_R(Size, N_{ÜE}) = \left(\left\lceil \frac{(Size-27)}{31} \right\rceil + 2 \right) \times N_{ÜE} \qquad (15)$$

jedoch linear zunimmt. Weiterführende Überlegungen zu der Problematik der Wahl der entsprechenden Parameter können beispielsweise den Ausführungen von *Dünkel et al.* [152] entnommen werden.

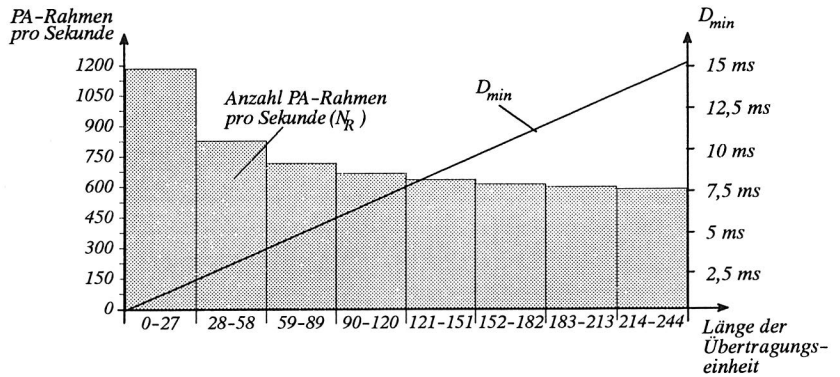


Bild 83: Abhängigkeit von D_{min} und N_R von der Länge der Übertragungseinheit bei f_{DS} = const.

Der Manager antwortet auf eine derartige Anfrage mit einer positiven oder negativen Quittung. Dabei darf eine negative Quittung nur dann gesendet werden, wenn den Anforderungen aufgrund nicht verfügbarer Bandbreite nicht entsprochen werden kann. Bei der Überprüfung der Verfügbarkeit der geforderten Bandbreite wird zuerst untersucht, ob eine periodische Anforderung für die übergebene Pufferplatzidentifikation bereits bearbeitet wird. Ist dies der Fall, so wird die Rate überprüft. Stimmen die bereits berücksichtigte und die angeforderte Rate überein, so wird eine Sperrung oder Freigabe der Einträge vorgenommen und eine positive Bestätigung gesendet. Stimmen die Raten nicht überein und sind die Einträge freigegeben, so wird eine negative Quittung gesendet. Betrifft die Anforderung eine aktuell nicht periodisch ausgetauschte Pufferplatzidentifikation und kann sie eingeplant werden, so wird die Einplanung vollzogen und eine positive Quittung gesendet. Kann die Einplanung aufgrund der aktuellen Belegung der Pollingtabelle nicht vorgenommen werden, so wird eine negative Quittung gesendet. Die Einplanungsstrategie ist beliebig wählbar, ist jedoch in allen als redundante BKS fungierenden Stationen einheitlich zu halten.

7.6 Synchronisation der verteilten Uhren

Für die Zeitstempelung von Variablen, zur Behandlung zeitbehafteter Daten im verteilten System und zur Synchronisation der Applikationen in den verteilten Stationen wird eine gemeinsame Zeitreferenz für alle angeschlossenen Stationen bereitgestellt.

7.6.1 Allgemeine Betrachtungen

Jede OSIRIS-Station verfügt über eine eigene lokale Uhr, auf deren aktuellen Stand vom Kommunikationssystem und von den Anwendungen aus über das OMP zugegriffen werden kann. Die Synchronisation aller lokalen Uhren wird mittels des zur Funktion des Systems notwendigen, gleich-

mäßigen Rahmentaktes (vgl. Bild 47) realisiert. Damit ist ohne zusätzliche Maßnahmen die Granularität G der Uhr direkt von der Rahmenrate f_R abhängig, deren Berechnung bereits in Formel 6 angegeben wurde. Für die Granularität ergibt sich folgende Berechnungsvorschrift:

$$G(t_R,\ t_{Lat}) = \frac{1}{f_R} = t_R + t_{Lat} \qquad (16)$$

7.6.2 Modell einer lokalen Uhr

Die lokale Uhr einer Station besteht aus zwei Bestandteilen: dem *lokalen Uhrenobjekt* (LCO, Local Clock Object), das die reale „Uhr" der Station, einen 32 Bit breiten Zähler, repräsentiert, und dem *Zeit-Dienste Objekt* (TSO, Time-Service Object), über das durch die Benutzer auf die Uhrzeit zugegriffen werden kann [153]. Das LCO bietet dem TSO an seiner Schnittstelle fünf Operationen an. Diese dienen dazu, den Zählerstand abzufragen („LCO-TimeRequest.req/res") bzw. einen neuen Zählerstand vorzugeben („LCO-TimeSet.req"), die Uhr zu initialisieren („LCO-Init.req") oder sie zu starten („LCO-Start.req"). Wird während des Laufes der Uhr eine neue Zeit geladen, die nicht mit der übereinstimmt, deren Wert durch den Stand des Zählers repräsentiert wird, so wird vom LCO mittels der Funktion „LCO-ExceptionReport" eine entsprechende Benachrichtigung an das TSO abgesetzt, die als Parameter den alten Zählerstand enthält.

Das TSO ruft nur dann Operationen des LCO auf, wenn es über die Benutzer- oder über die Managementschnittstelle dazu aufgefordert wird. Aufrufe über die Managementschnittstelle dienen der Synchronisation aller Stationen und werden von dem für die Synchronisation aller Uhren zuständigen Manager vorgenommen. An der Benutzerschnittstelle stellt das TSO eine Operation zur Abfrage der aktuellen Uhrzeit zur Verfügung („SM-TimeRequest", vgl. Bild 84).

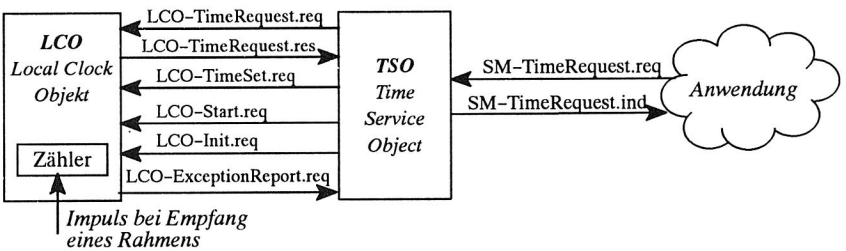


Bild 84: Struktur der stationsinternen Uhr

Das Verhalten des LCO ist wie folgt spezifiziert: Nach dem Einschalten bzw. nach einer Initialisierung befindet sich das LCO im Zustand „Unsynchronisiert" und es werden solange keine Rahmen gezählt, bis die Aufforderung zum Start entgegengenommen wurde. In beiden Fällen wird als aktueller Zählerstand der Wert „0" geführt. Lesezugriffe auf den Zähler werden unter Angabe des Grundes („Zähler nicht aktiv") abgewiesen. Nach der Aufforderung zum Starten des Zählers inkrementiert das LCO bei Eintreffen des Impulses, der den Beginn des Empfangs eines Rahmens anzeigt, jeweils den Zähler. Es besteht jederzeit die Möglichkeit, ohne Anhalten der Uhr einen beliebigen gültigen Wert in den Zähler zu laden. Diese Funktion dient dazu, die Synchronisation der Stationen im System zu überprüfen und neu in das System aufgenommene Stationen zu synchronisieren. Das funktionale Verhalten des LCO wird durch das in Bild 85 dargestellte Zustandsübergangsdiagramm verdeutlicht.

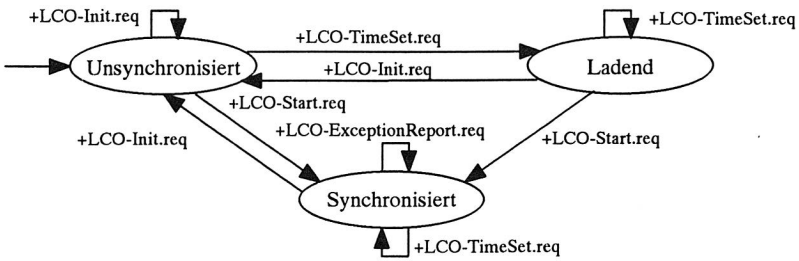


Bild 85: Zustandsübergänge des LCO

In OSIRIS kommt es, bedingt durch die nicht zu vernachlässigende Verzögerung von Nachrichten durch die Stationen und durch die Nutzung des Rahmentaktes für die Synchronisation, zu einer Verschiebung der Zeitmaßstäbe zwischen den einzelnen Stationen, die bei der Uhrzeitsynchronisation berücksichtigt werden muß.

Seien die n Stationen eines OSIRIS-Netzwerkes im Uhrzeigersinn numeriert und habe die BKS die Nummer „0". Seien $t_{Über}(i,j)$ die Übertragungszeit zwischen den Stationen i und j und $t_{Lat}(i)$ die Latenzzeit eines Rahmens in der Station i. Dann ergibt sich die Verzögerung zwischen dem Aussenden eines Rahmens durch die BKS und der Ankunft in der Station $k \in [1,n]$ über den Bus A, die *kumulierte Latenzzeit* $t_{KLat,A}(k)$ wie folgt:

$$t_{KLat,A}(k) = \sum_{i=1}^{k-1} \left(t_{Über}(i-1, i) + t_{Lat}(i)\right) \qquad (17)$$

Die Signalübertragungszeit in einem verdrillten Leitungspaar liegt bei Werten von 135 Ω für den Wellenwiderstand und einer Kabelkapazität von 30 pF/m, die der Spezifikation von PROFIBUS-DP [112] entstammen, bei ca. 82,36 % der Lichtgeschwindigkeit. Die grundlegenden Formeln werden z.B. von *Pimentel* [154] vorgestellt und an Beispielen erläutert. In einem System mit einer Gesamtkabellänge von 250 m beträgt die kumulierte Verzögerungszeit der Nachricht durch die Übertragung ca. 1,01 µs und kann damit hier vernachlässigt werden. Es ergibt sich folgende, vereinfachte Formel:

$$t_{KLat,A}(k) \approx (k-1) \times t_{Lat} \qquad (18)$$

Die Verzögerung der Rahmen zwischen der Generierung durch die BKS und dem Empfang bei der Station k über Bus B ergibt entsprechend, wird aber hier nicht weiter betrachtet.

Wird die kumulierte Latenzzeit nicht berücksichtigt, kann es dazu kommen, daß ein Ereignis, das von mehreren Stationen beobachtet wird, jeweils mit unterschiedlichen Zeitstempeln $\widehat{C}_{Station}$ versehen wird. In dem in Bild 86 dargestellten Beispiel werden dem Zeitpunkt E_1, zu dem ein Ereignis in Realzeit eintritt, gleiche Zeitstempel zugewiesen ($\widehat{C}_0(E_1) = \widehat{C}_1(E_1) = \widehat{C}_2(E_1) = 1$), dem folgenden Ereignis, daß zum Zeitpunkt E_2 eintritt, aber nicht ($\widehat{C}_0(E_2) = 5 \neq \widehat{C}_1(E_2) = \widehat{C}_2(E_2) = 4$). Eine entsprechende Korrektur wird durch das TSO vorgenommen. Dazu steht jeder Station der Wert für die kumulierte Latenzzeit der zwischen ihr und der BKS befindlichen Stationen auf dem zur Synchronisation eingesetzten Bus zur Verfügung. Diese Information kann in einer beliebigen Station durch die Anzahl der in der Erreichbarkeitstabelle für den Bus B aufgeführten Stationen

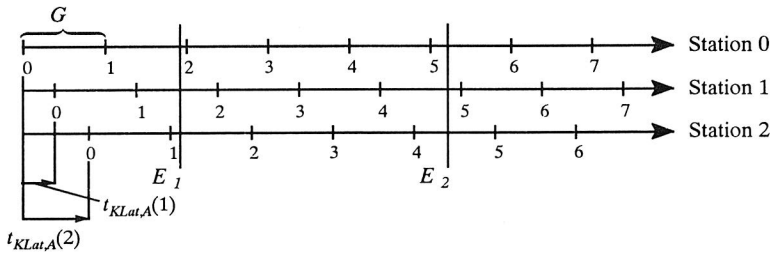


Bild 86: Auswirkung der Rahmenverzögerung ohne Kompensation

ermittelt werden. Damit wird auch eine dynamische Anpassung von $t_{KLat,A}(k)$ bei Änderungen der Konfiguration ermöglicht.

Die Korrektur besteht darin, bei einer Leseanfrage vor der Rückgabe des Wertes den vom LCO gelieferten Wert $\hat{C}_k(t)$ mit einem Korrekturwert zu beaufschlagen. Dieser Korrekturwert δ wird für eine beliebige Station k wie folgt gebildet:

$$\delta_k = \begin{cases} \left\lfloor \frac{t_{KLat}(k)}{(t_R + t_{Lat})} \right\rfloor & \text{für } t_{KLat,A}(k) \bmod G < 0,5 \\ \left\lfloor \frac{t_{KLat}(k)}{(t_R + t_{Lat})} \right\rfloor + G & \text{für } t_{KLat,A}(k) \bmod G \geq 0,5 \end{cases} \qquad (19)$$

Zur Abfrage der Zeit bietet das TSO bietet an seiner Schnittstelle zum Benutzer die Operation `SM-TIME-REQUEST` an. Der auf Anfrage der Anwendungen vom TSO zurückgelieferte Wert $\tilde{C}_k(t)$ wird wie folgt gebildet:

$$\tilde{C}_k(t) = \hat{C}_k(t) + \delta_k \qquad (20)$$

Befindet sich das LCO nicht im Zustand „Synchronisiert", so wird eine entsprechende Fehlermeldung an den Dienstbenutzer des TSO übergeben.

Für gewählte Rahmenraten ist ein Bezug zur Realzeit herstellbar. Dabei kommt es in direkter Abhängigkeit von f_R und der Zählerbreite von 32 Bit zu Überläufen des Zählers und zu dessen Rücksetzen modulo 2^{32}. Die bis zum Rücksetzen erreichbaren Zeitspannen D_{mod} sind in Tabelle 35 für $L_Ü$=2 Bit und t_A=24 Bitzeiten und für ausgewählte, in Feldkommunikationssystemen häufig verwendete, Datenübertragungsraten in Realzeit dargestellt. Für die Angabe von D_{mod} wurde eine Form gewählt, in der die Anzahl der Tage, Stunden, Minuten, Sekunden und Hundertstelsekunden jeweils durch Doppelpunkte getrennt sind.

Par. $f_Ü$	31250 bps	500000 bps	1 Mbps	2,5 Mbps	5 Mbps
t_R	$11.52 * 10^{-3}$ s	$0,72 * 10^{-3}$ s	$0,36 * 10^{-3}$ s	$0,144 * 10^{-3}$ s	$0,072 * 10^{-3}$ s
G	$12,288 * 10^{-3}$ s	$0,768 * 10^{-3}$ s	$0,384 * 10^{-3}$ s	$0,1536 * 10^{-3}$ s	$0,0768 * 10^{-3}$ s
f_R	$81,38 * s^{-1}$	$1302,08\overline{33} * s^{-1}$	$2604,1\overline{66} * s^{-1}$	$6510,41\overline{66} * s^{-1}$	$13020,8\overline{33} * s^{-1}$
D_{mod}	610:20:9:18:12	38:4:15:34:88	19:2:7:47:77	7:15:15:6:98	3:19:37:44:49

Tabelle 35: Charakteristische Parameter für die Uhrzeitsynchronisation

Damit tritt auch bei einer Datenübertragungsrate von 5 Mbps ein Überlauf und Zurücksetzen des Zählers erst größenordnungsmäßig im Bereich von Tagen auf, so daß eine weitere zeitliche Zuordnung, z.B. zur politischen Zeit, ggf. durch auf das Feldkommunikationssystem aufsetzende Applikationen vorgenommen werden kann.

7.6.3 Netzweite Synchronisation

Um eine globale Synchronisation aller LCOs zu gewährleisten übernimmt die BKS die Rolle der die Synchronisation der Uhren steuernden Station. Dafür stehen ihr drei unbestätigte Managementdienste zur Verfügung: `SM-STARTCLOCK` zum Starten der verteilten Uhren mit dem bei der lokalen Uhr vorhandenen Zählerstand und `SM-INITCLOCK` zum Zurücksetzen der verteilten Uhren. Das Setzen der Uhr kann mittels des unbestätigten Dienstes `SM-SETCLOCK` geschehen. Die Dienste `SM-STARTCLOCK` und `SM-INITCLOCK` sind parameterlos. Der Dienst `SM-SETCLOCK` benötigt als Aufrufparameter die Angabe der zu setztenden Zeit.

Alle Dienste werden grundsätzlich hochprior im Broadcast gesendet und in der Station an das TSO übergeben, das den Aufruf der entsprechenden Operationen vornimmt. Eine implizite Sicherstellung, daß die Information in allen Stationen angekommen ist, kann durch eine Überprüfung des Empfanges des gesendeten Managementrahmens durch die BKS vorgenommen werden.

Wie aus Bild 85 hervorgeht, wird eine eintreffende Aufforderung zum Laden einer neuen Zeit auch dann berücksichtigt, wenn das LCO sich im Zustand „Synchronisiert" befindet. Sind bei der Übertragung der Nachricht keine Fehler aufgetreten und arbeitet der Zähler fehlerfrei, so stimmen der Zählerstand und der zu ladende Wert überein. Um Übertragungsfehler mit großer Wahrscheinlichkeit ausschließen zu können, wird der zu ladende Wert zweifach übertragen. Wird ein Übertragungsfehler festgestellt, so werden keine Modifikation am lokalen Zähler vorgenommen. Anderenfalls wird der Wert in den lokalen Zähler geladen. Wird dabei vom LCO eine Differenz zwischen dem aktuellen Wert und dem zu ladenden Wert festgestellt, so wird eine entsprechende Mitteilung an das TSO mittels des Dienstes `LCO-ExceptionReport` abgesetzt. Das TSO leitet diese lokale Information als Management-Information an die BKS weiter. Dazu wird der unbestätigte Management-Dienst `SM-FAULT-NOTIFICATION` genutzt, der im Parameter „Additional Detail" den aktuellen Wert der lokalen Uhr und den neu geladenen Wert enthält. In der BKS können die eintreffenden Informationen archiviert werden.

Zusätzlich zu dieser Möglichkeit einer sporadischen Uhrzeitsynchronisation wird in den periodisch ausgetauschten Management-Rahmen der Stand des Zählers der Uhr der Buskopfstation übertragen. Der Abgleich der lokalen Uhr einer Station erfolgt nach dem oben vorgestellten Verfahren.

7.6.4 Betrachtungen zur Genauigkeit

Sind alle Uhren synchronisiert, so ist die maximale Abweichung zweier beliebiger Uhren gleich der Granularität G. Nach *Hofmann* [102] sind damit Ereignisse nur dann in ihrer Reihenfolge sicher auflösbar, wenn sie in einem zeitlichen Abstand von mindestens 2 G auftreten. Für ausgewählte Bitübertragungsraten, bei den auch für die Darstellungen in Tabelle 35 beispielhaft gewählten Parameter, beträgt die Granularität der Uhr bei 1 Mbps 384 µs, bei 2,5 Mbps 154 µs und bei 5 Mbps 72 µs. Damit können Ereignisse sicher bezüglich ihrer zeitlichen Reihenfolge aufgelöst werden, wenn sie in einem zeitlichen Abstand von 768 µs, 308 µs bzw. 144 µs auftreten.

Von der Anwendung können vom LCO zudem Informationen über die Genauigkeit der Synchronisation der lokalen Uhr angefordert werden. Die lokale Feststellung der Synchronisationsgenauig-

keit ist dabei nicht unproblematisch und wird in OSIRIS wie folgt behandelt: Aufgrund des dargestellten Verfahrens haben alle Stationen, sofern sie sich im Zustand „Synchronisiert" befinden, einen Zählerstand und damit eine Uhrzeit, die sich zwischen zwei beliebigen Stationen maximal um G unterscheidet. Damit reicht zur Angabe der Güte der Synchronisation der lokalen Uhr die Information aus, ob das LCO sich im Zustand „Synchronisiert" befindet oder nicht. Befindet es sich nicht in diesem Zustand, so kann keine quantitative Aussage über die Güte der Synchronisation gemacht werden. Alle Betrachtungen setzen zudem ein fehlerfreies Verhalten des LCO und des TSO voraus.

7.7 Leistungsmanagement

Das Leistungsmanagement in OSIRIS dient vornehmlich der Sammlung statistischer Informationen zum Zwecke der späteren Auswertung. Die Informationen werden in Zählern gehalten. Im einzelnen werden dabei betrachtet:

- in OMP:
  - Anzahl empfangener bzw. gesendeter Protokolldateneinheiten (OMP-NUM-RPDU, OMP-NUM-SPDU)
- in RTSIMS:
  - Anzahl lokal bzw. entfernt verursachter Zurückweisungen von Protokolldateneinheiten (RT-REJ-LOC, RT-REJ-REM)
  - Anzahl empfangener Protokolldateneinheiten (RT-NUM-RPDU)
  - Anzahl gesendeter Protokolldateneinheiten (RT-NUM-SPDU)
- im LLI:
  - Anzahl der eingegangenen Verbindungsauf- bzw. -abbauwünsche, die angenommen bzw. zurückgewiesen wurden (LLI-INCON-OK, LLI-INCON-REJ, LLI-INDIS-OK, LLI-INDIS-OK)
  - Anzahl der gesendeten Verbindungsauf- bzw. abbauwünsche, die angenommen bzw. zurückgewiesen wurden (LLI-OUTCON-OK, LLI-OUTCON-REF, LLI-OUTCON-OK, LLI-OUTDIS-REF)
  - Anzahl der gesendeten Protokolldateneinheiten (LLI-NUM-PDU)
- im DRSLP:
  - Anzahl angeforderter Variablenwerte (DR-VAL-REQ)
  - Anzahl gesendeter Variablenwerte (DR-VAL-SENT)
  - Anzahl der Wiederholungen bei gesicherter Nachrichtenübertragung (DR-RETRY)
  - Anzahl gesendeter Nachrichten-Protokolldateneinheiten (DR-NUM-MPDU)
  - Anzahl nicht erfüllbarer Echtzeit-Nachrichtenübertragungen (DR-RTMSG-REF)
  - Anzahl fehlerhaft eingetroffener Protokolldateneinheiten (DR-CORR-PDU)
- im DQDFB:
  - Anzahl der aufgrund von ungültigen Zieladressen zurückgewiesenen Protokolldateneinheiten (DQ-INV-DA)
  - Anzahl eingetroffener Oktette (DQ-OCT-REC)
  - Anzahl gesendeter Oktette (DQ-OCT-SENT)

Die in Klammern nachgestellten Bezeichnungen dienen als symbolische Namen für die Zähler, die durch die jeweilige Protokollinstanz geführt werden. Diese Variable des Managements können mittels der definierten Dienste gelesen, geschrieben und zurückgesetzt werden.

7.8 Zusammenfassung der wichtigsten Eigenschaften des Managements

In OSIRIS ist Funktionalität für die Verwaltung von Anwendungen, der Weiterleitung von Fehlermeldungen, der Konfiguration des Systems, der Erfassung von Systemgrößen zur Beurteilung der Leistung und der Synchronisation der verteilten Uhren im Systemmanagement definiert.

Im Hinblick auf das Konfigurationsmanagement kann konstatiert werden, daß die diesbezüglich bei DQDB vorhandenen Probleme, wie sie z.B. von *Spaniol* [155] beschrieben werden, beim Entwurf von OSIRIS berücksichtigt wurden. Dazu zählt insbesondere die Festlegung von Management-Funktionalität zur dynamischen Aktualisierung der Systemvariablen, z.B. der Erreichbarkeitstabellen und der RQ- und CD-Zähler. Die Möglichkeit, die zur Steuerung des periodischen Datenaustausches verwendeten Pollingtabellen dynamisch zu modifiziern, sorgt für die benötigte Flexibilität des Systems. Die Synchronisation der verteilten Uhren basiert auf dem in OSIRIS gegebenen gleichmäßigen Takt, mit dem Übertragungsrahmen zwischen den Stationen ausgetauscht werden. Sie bietet eine von der Systemkonfiguration abhängige Granularität und eine maximale Abweichung der lokalen Uhrzeit zweier Stationen in Höhe der Granularität der Uhr.

7.9 Bewertung

OSIRIS wurde anhand der aufgestellten Anforderungen konzipiert. Nachfolgend werden die wesentlichen Eigenschaften des Systems kurz resümiert, mit denen die einzelnen Anforderungen erfüllt werden.

ANFORDERUNG ***EZ-1: Rechtzeitigkeit***

OSIRIS stellt Echtzeit-Kommunikationsdienste für den Austausch von Variablen und Nachrichten zur Verfügung. Der deterministische Austausch von Variablen wird unter der Kontrolle der Buskopfstation durch Mechanismen der Protokolle der Sicherungsschicht erbracht. Für das Senden von unbestätigten Nachrichten können in der sendenden Station zeitliche Obergrenzen für den Sendevorgang angegeben werden.

ANFORDERUNG ***EZ-2: Berücksichtigung von Echtzeit- und Nicht-Echtzeit-Kommunikationsanforderungen***

Neben den Echtzeit-Kommunikationsdiensten stehen Dienste für den aperiodischen Austausch von Variablen und Nachrichten zur Verfügung. Für den Austausch von Nachrichten besteht dabei die Möglichkeit der Wahl zwischen gesicherter und ungesicherter Übertragung.

ANFORDERUNG ***EZ-3: Vorhersehbarkeit***

Das Verhalten von OSIRIS auf die Einlastung von Kommunikationsaufträgen ist vorhersehbar. Bezüglich des periodischen Austausches von Variablen kann eine Überlastung durch die deterministische Steuerung dieser Übertragungsart anhand der Pollingtabelle nicht auftreten. Beim Austausch von Nachrichten unter Echtzeitbedingungen ist die pro Station dafür zur Verfügung stehende Bandbreite bekannt. Anhand dieser Information und der Anzahl bereits berücksichtigter Anforderungen wird festgestellt, ob ein neuer Übertragungsauftrag noch rechtzeitig ausgeführt werden kann. Ist dies nicht der Fall, so erfolgt eine Rückweisung des Auftrages.

ANFORDERUNG ***EZ-4: Zuverlässigkeit***

Zur Erhöhung der Zuverlässigkeit eines OSIRIS-Systems ist Redundanz bezüglich der Funktion der Buskopfstation vorgesehen. Alle Stationen sind aktiv an das Medium angekoppelt, können jedoch bei Fehlfunktion vollständig vom System entkoppelt werden, womit nach *Vogt et al.* [156]

ebenfalls eine Erhöhung der Zuverlässigkeit verbunden ist. Die Sicherung einzelner, zwischen den Stationen übertragener, Dateneinheiten geschieht mittels einer Blockprüfsumme.

Festgestellte Fehlfunktionen, z.B. eine fehlerhafte Uhrzeitsynchronisation oder ein Überlauf eines Zählers, werden über das Systemmanagement an den zentralen Manager-Prozeß weitergeleitet und können dort weiter ausgewertet werden.

ANFORDERUNG ***EZ-5: Verarbeitung von Echtzeit-Prozeßvariablen***

Neben den regulären Prozeßvariablen sind in RTSIMS *Echtzeitvariable* definiert. Sie werden über die verteilte Datenbasis ausgetauscht. Der schreibende und der lesende Zugriff auf Echtzeitvariable ist mittels gesondert definierter Dienste möglich, die auf die temporale Gültigkeit der Werte – unter Beachtung der Güte der Uhrzeitsynchronisation in den an der Erzeugung bzw. Konsumierung beteiligten Stationen – Rücksicht nehmen. Der lesende Zugriff auf den Wert zeitkritischer Variable ist auch mittels des regulären Lesedienstes READ möglich (Polymorphismus). In diesem Fall erfolgt keine Auswertung der besonderen Attribute von zeitkritischen Variablen.

ANFORDERUNG ***EZ-6:*** **Unterstützung einer verteilten Uhr**

Eine verteilte, synchronisierte Uhr wird in OSIRIS basierend auf dem Takt des Austausches der Übertragungsrahmen realisiert. Die Granularität der Uhr und die maximale Abweichung zweier beliebiger Uhren sind in Abhängigkeit von der Rahmenrate angebbar. Eine periodische Überprüfung der Korrektheit der lokal gehaltenen Zeit sowie die Synchronisation neu in das System aufgenommener Stationen sind berücksichtigt. Die von *Lamport* aufgestellten Bedingungen für physikalische Uhren werden erfüllt.

ANFORDERUNG ***VS-1: Benutzergesteuerte Prioritätenvergabe***

Auf Ebene der Sicherungsschicht werden für die aperiodischen Übertragung von Daten zwei verschiedene Prioritätsstufen bereitgestellt, die auch der Anwendung bei allen entfernt wirkenden Diensten zur Verfügung stehen.

ANFORDERUNG ***VS-2: Synchronisation von Applikationen***

Die Synchronisation von Applikationen kann in OSIRIS auf mehrere Arten geschehen:

- Zeitgesteuert durch den Scheduler,
- ereignisgesteuert durch das Versenden von Ereignismeldungen und Einplanung durch den Scheduler und
- zeitgesteuert durch die festgelegte Gültigkeit von Echtzeitvariablen

und weist somit die geforderte große Flexibilität und Ausdrucksfähigkeit auf.

ANFORDERUNG ***VS-3: Scheduling des Kommunikationssystems***

Die Zuteilung von Kommunikationsbandbreite erfolgt hybrid durch die Reservierung von Übertragungsrahmen und durch das verteilte, faire Zugriffsverfahren auf die zum Zwecke der aperiodischen Datenübertragung dienenden Übertragungsrahmen. Die Reservierung von Kommunikationsbandbreite kann statisch oder dynamisch erfolgen. Die statische Reservierung wird mittels vordefinierter Einträge in der Pollingtabelle vorgenommen. Zur Modifikation der Pollingtabelle zur Laufzeit des Systems stehen Dienste des Systemmanagements zur Verfügung.

ANFORDERUNG ***VS-4: Breites Spektrum an PDU-Sequenzen***

Alle vier geforderten PDU-Sequenzen werden unterstützt. Die Sequenzen vom Typ 1 und Typ 2 werden beim unbestätigten Senden einer Nachricht verwendet; die vom Typ 3 findet beim gesi-

cherten Datenaustausch Anwendung. Zur Wahrung der Konsistenz der verteilten Datenbasis wird die Sequenz vom Typ 4 verwendet. Die Anforderung wird durch eine Protokolldateneinheit vom Typ „Publish-Request" und die Quittung bzw. Antwort durch eine vom Typ „Publish-Value" gebildet.

ANFORDERUNG VS-5: Autonomie

Eine gegenüber den anderen betrachteten Systemen größere Autonomie wird in OSIRIS durch die Einführung der zeit- und ereignisgesteuerten Ausführung von Programminstanzen erreicht. Die Möglichkeit, die für die Steuerung der Programmausführung zu benutzenden Tabellen dynamisch zu erzeugen und ihre Verwendung anzustoßen bedeutet dabei ein hohes Maß an Flexibilität.

Die Feststellung von erfüllten Ereignisbedingungen und die Initiierung zu ergreifender Maßnahmen wird durch das Kommunikationsprotokoll bewerkstelligt und wirkt über den Scheduler direkt auf die Ausführungsreihenfolge der Programminstanzen.

ANFORDERUNG VS-6: Komfortable Kommunikationsdienste

Für die Anwendungsschicht von OSIRIS ist mit RTSIMS ein Protokoll mit komfortablen Kommuniktionsdiensten zur Steuerung von Fertigungsgeräten, Multimedia-Aufnahmegeräten und zum Zugriff auf die verteilte Datenbasis definiert. Die Definition dieses Protokolls basiert auf dem Prinzip der Objektorientierung, wodurch sich bei äquivalenter Funktionalität eine im Vergleich mit anderen Protokollen geringere Anzahl von Diensten ergibt.

Anders als in FMS und MMS liegt dem Austausch von Variablen in RTSIMS grundsätzlich die verteilte Datenbasis zugrunde. An der Benutzerschnittstelle ist die Berücksichtigung der unterschiedlichen Kommunikationsmodelle weitgehend transparent. Die Feststellung von Ereignissen wird im Gegensatz zu PROFIBUS durch das Kommunikationsprotokoll der Anwendungsschicht vorgenommen.

ANFORDERUNG VS-7: Berücksichtigung von ereignis- und zustandsgesteuerter Kommunikation

Die Berücksichtigung ereignis- und zustandsgesteuerter Kommunikation ist integraler Bestandteil des Konzeptes von OSIRIS. Die zustandsgesteuerte Kommunikation wird durch die deterministischen, zeitgesteuerten Mechanismen von DQDFB und DRSLP realisiert. Eine Anbindung der asynchron dazu laufenden Applikationen erfolgt über Dienstankündigungen beim Senden bzw. Empfang von Dateneinheiten. Die ereignisgesteuerte, aperiodische Kommunikation wird sowohl bezüglich des Austausches von Nachrichten als auch von Variablen unterstützt.

ANFORDERUNG VS-8: Unterstützung von Replikationsmechanismen für Daten

Wie auch das FIP-System, so bietet OSIRIS einen Replikationsmechanismus für Prozeßvariable an, der durch die Protokollinstanzen der Sicherungsschicht abgewickelt wird. Die a priori zu diesem Datenaustausch asynchron ablaufenden Applikationen können mittels definierter Dienstankündigungen selektiv über den Fortgang des Austausches informiert werden.

ANFORDERUNG MM-1: Übertragung kontinuierlicher und diskreter Medien

Die Übertragung beider Medientypen wird unterstützt. Zur Übertragung von Daten, die zu Datenströmen kontinuierlicher Medien gehören, wird der Mechanismus des periodischen Datenaustausches verwendet. Damit kann ein isochroner Austausch von Dateneinheiten gewährleistet werden. Durch dynamische Bandbreitenreservierung ist eine Anpassung an sich ändernde Anforderungen, z.B. nach einer Modifikation der Kodierungsgüte, möglich. Die notwendige Verwaltung der

Gruppe der als Senken fungierenden Stationen wird durch das Protokoll der Anwendungsschicht erbracht.

Zur Übertragung diskreter Medien wird die aperiodische, nachrichtenorientierte Kommunikation von OSIRIS verwendet. Dabei kommen die Mechanismen des Domain-Managements zum Einsatz.

ANFORDERUNG MM-2: Unterstützung von Aufnahmesystemen durch das Kommunikationsprotokoll

OSIRIS beinhaltet im Protokoll der Anwendungsschicht Objekttypen zur abstrakten Modellierung von Aufnahmesystemen zusammen mit den auf die Instanzen dieser Objekttypen wirkenden Operationen, die im realen System als Dienste für den Client verfügbar sind. Die bereitgestellten Operationen dienen der Steuerung des Aufnahmesystems und werden über Nachrichtenübertragungsdienste abgewickelt. Sie beinhalten Funktionen zum Starten und Beenden der Aufnahme, zur Positionierung und Einstellung des Systems, zur Abfrage von dessen Status sowie zur Neuaushandlung von Parametern für das Kodierungsverfahren. Der Server verfügt über Mechanismen zur Verwaltung der Gruppe der zu einem beliebigen Zeitpunkt als Senken fungierenden Client-Applikationen.

ANFORDERUNG MM-3: Synchronisation von Multimedia-Datenströmen

Die Synchronisation von Multimedia-Datenströmen wird in OSIRIS durch die Verwendung eines getakteten Übertragungssystems mit fest reservierten Rahmen realisiert. Aufgrund der Beschränkung der Netztopologie auf ein Segment kann die Synchronisation von aus verschiedenen Datenquellen eintreffenden Datenströmen durch die Applikationen in den als Senken fungierenden Stationen vorgenommen werden. Der gleichzeitge Start der Aufnahmensysteme kann zeit- oder ereignisgesteuert vorgenommen werden.

7.10 Zusammenfassung und ergänzende Bemerkungen

In den Kapiteln fünf bis sieben wurde mit OSIRIS ein Systemkonzept vorgestellt, das anhand der aus der Entwicklung der Steuerungstechnik und der geforderten Unterstützung für multimediale Applikationen ableitbaren Anforderungen an Feldkommunikationssysteme konzipiert wurde. Die vorgestellte Systemarchitektur umfaßt insbesondere die Protokolle der Sicherungs- und Anwendungsschicht sowie das Systemmanagement und ist in der Lage, die in Kapitel 3 dargestellten Anforderungen vollständig zu erfüllen.

Zur Validierung des Konzeptes wurde die Datenübertragung mittels DQDFB auf Basis einfacher serieller Schnittstellen bzw. über die Links von Transputern – jeweils mit Konfigurationsmanagament – ebenso realisiert, wie ein rudimentäres Werkzeug zur Erzeugung von Pollingtabellen. Für DRSLP liegt eine formale Spezifikation in *Estelle* [157] vor. Des weiteren wurden die Bestandteile von RTSIMS, die eine Steuerung einer virtuellen Kamera für Stand- bzw. Bewegtbilder zulassen, auf Basis von PROFIBUS implementiert. Die Methodik der Behandlung von Echtzeitvariablen und die Integration der Unterstützung des Ereignismanagements in das Kommunikationssystem wurden bereits im OLCHFA-System erprobt.

8 Zusammenfassung und Ausblick

Die Entwicklung der Steuerungstechnik steht unter dem Zeichen der Dezentralisierung von Funktionen, die durch die Entwicklungen in der Mikroelektronik mit der Verfügbarkeit preiswerter und leistungsfähiger Komponenten zur Erbringung von Verarbeitungskapazität ermöglicht werden. Die Aufteilung der gesamten Funktionalität, die zur Steuerung eines technischen Prozesses notwendig ist, führt zu der Schaffung von teilweise autonomen Einheiten, die durch Erbringung ihrer Funktionen im Zusammenwirken zur Erfüllung der Gesamtaufgabe beitragen. Dazu ist eine Koordination der einzelnen funktionalen Einheiten notwendig, die über das Kommunikationssystem schritthaltend mit dem Ablauf des technischen Prozesses erbracht werden muß. An das einzusetzende Kommunikationssystem stellen sich dadurch die in dieser Arbeit dargestellten diesbezüglichen Anforderungen.

Trotz umfangreichen Rechnereinsatzes bleibt der Mensch zumindest als kontrollierende Instanz unabdingbar. Diesbezüglich ist die Bereitstellung der Informationen, die benötigt werden, um auf den Zustand des technischen Prozesses schließen zu können, eine wesentliche Komponente der Bedien- und Beobachtungstechnologie, die sich zum treibenden Faktor des Einsatzes von Multimedia-Technologie in den prozeßnahen Bereichen der rechnerintegrierten Fertigung entwickelt hat. Die Verfügbarkeit von visueller und auditiver Information, die direkt aus dem Prozeß abgegriffen und in Echtzeit an den Anlagenbediener übertragen wird, erleichtert diesem die frühzeitige und richtige Erkennung von sich anbahnenden oder bereits eingetretenen Störungen, kann aber auch z.B. im Bereich der Qualitätssicherung und der Diagnose von großer Bedeutung sein. Ein Vorliegen all dieser Informationen in digitaler Form ermöglicht auch die aufwandsarme Ferndiagnose und damit den Zugriff auf vor Ort nicht verfügbares Expertenwissen.

Bei der Definition der Anforderungen wurde die Entwicklung in Richtung des Einsatzes von Multimedia-Applikationen dahingehend berücksichtigt, daß die Integration der Steuerung der Aufnahmesysteme, z.B. positionierbare Kameras oder Mikrofone, und die Bereitstellung adäquater Kommunikationsmechanismen zum Austausch von zu kontinuierlichen Datenströmen gehörenden Dateneinheiten und zur Synchronisation von derartigen Datenströmen in den Forderungskatalog aufgenommen wurden. Die Charakteristik von Feldkommunikationssystemen, die Eigenschaften der Kodierungsverfahren und die Anforderungen der Anwendungen stellen dabei die wesentlichen Faktoren bei der durchgeführten Motivation der Menge geeigneter Verfahren für die Kodierung dar.

Die Bewertung der im europäischen Raum standardisierten Feldkommunikationssysteme PROFIBUS und FIP führt zu dem Ergebnis, daß diese die dargestellten Anforderungen nur teilweise und – zusammenfassend betrachtet – in einem nicht ausreichendem Maß erfüllen. Beide Systeme weisen, trotz des im wesentlichen gleichen anvisierten Anwendungsbereichs, aufgrund ihrer grundlegend verschiedenen Systemkonzepte in unterschiedlichen Bereichen Stärken und Schwächen auf.

Als Versuch, ein vorhandenes System konzeptionell und auch in dessen Implementierung weiter zu entwickeln, kann das OLCHFA-System betrachtet werden. In dem zugrundeliegenden Projekt ging es unter anderem darum, das FIP-System um eine synchronisierte Uhrzeit und – auf Basis der damit verfügbaren Zeitinformationen – um die Mechanismen zur Behandlung zeitbehafteter Daten zu erweitern. Die entstandenen Konzepte wurden in einer Implementierung validiert. Dabei trat die Problematik der Konfigurierung derartig komplexer Systeme deutlich zu Tage. Hier konnte durch angepaßte Werkzeuge beispielhaft gezeigt werden, wie Mechanismen der Uhrzeitsynchronisation, der Zeitbehaftung von Daten und der durch das Kommunikationssystem vorzu-

nehmenden Behandlung von Ereignissen bedienerfreundlich und komfortabel konfiguriert werden können.

Die Integration von Multimedia-Datenaustausch wurde am Beispiel des PROFIBUS-Systems unter Inkaufnahme der zu erwartenden Beschränkungen durchgeführt. Dabei wurde die Übertragung von Standbildern als diskrete Medien und von Video- und Audiodatenströmen als kontinuierliche Medien realisiert, die z.B. für Applikationen des Bedienens und Beobachtens genutzt werden können.

Es wurde in Anbetracht der bei der Analyse vorhandener Systeme und der bei deren Erweiterung und Nutzung erzielten Ergebnisse deutlich, daß eine vollständige Erfüllung der aufgestellten Anforderungen deren Berücksichtigung bereits in der Phase der Konzepterstellung für ein Feldkommunikationssystem verlangt. Aufbauend auf einem getakteten Übertragungssystem als Basis eines neuen, aufwandsarmen Synchronisationsverfahrens für verteilte Uhren und mit einem entsprechenden Medienzugriffsverfahren werden in dem vorgestellten System *OSIRIS* Mechanismen der verteilten Datenbasis, der nachrichtenorientierten Kommunikation und des Austausches von Multimedia-Datenströmen definiert. Die Protokolle der Sicherungsschicht wickeln dabei in Kooperation mit der zentralen Station des Systems, der Buskopfstation, alle die Aktionen autonom ab, die den periodischen Austausch von Informationen – gleich welchen Typs – betreffen. Damit wird eine Entlastung der Protokolle der Anwendungsschicht und auch der Anwendungen erreicht. Die Berücksichtigung von Echtzeitanforderungen an Kommunikationsdienste ist dabei sowohl für periodisch als auch für aperiodisch anfallende Anforderungen der Anwendung gegeben.

In der Anwendungsschicht von OSIRIS ist mit RTSIMS erstmals ein Protokoll definiert, das komfortable Dienste zur Steuerung von Fertigungsgeräten und Aufnahmesystemen für multimediale Datenströme und diskrete Medien anbietet. Dabei findet eine im Bereich der Feldkommunikationssysteme neue, für den Benutzer weitgehend transparente Integration der Kommunikationsmodelle des Austausches von Daten über die verteilte Datenbasis und der nachrichtenorientierten Kommunikation statt. Die geforderte Mehrpunktkommunikation für den Austausch von Multimedia-Daten mittels periodischer Übertragungsdienste wird ebenso berücksichtigt, wie der gegenseitige Ausschluß beim Zugriff auf wesentliche Steuerungsfunktionen der Aufnahmegeräte.

Die Kommunikationsfunktionalität zum Austausch von Nutzdaten zwischen Anwendungen zur Steuerung und Überwachung technischer Prozesse wird durch ein definiertes Systemmanagement ergänzt, das unter anderem eine dynamische Rekonfigurierung des Systems und die Synchronisation der verteilten Uhren zuläßt.

Mit der anvisierten vollständigen Realisierung des in dieser Arbeit neu vorgestellten Systems OSIRIS steht für den prozeßnahen Bereich der rechnerintegrierten Fertigung ein Feldkommunikationssystem zur Verfügung, daß allen Anforderungen nach Unterstützung von Autonomie und Echtzeitkommunikation für die Prozeßsteuerung und der Unterstützung für multimediale Applikationen, z.B. für das Bedienen und Beobachten technischer Prozesse, weitaus besser gerecht wird, als die heute verfügbaren Systeme.

Aber auch einzelne Bestandteile, insbesondere das Protokoll RTSIMS der Anwendungsschicht oder Teile davon, lassen sich in aufbauenden Arbeiten auf andere Feldkommunikationssysteme abbilden und in innovativen Steuerungskonzepten einsetzen. Bei der Fortsetzung der Arbeiten bezüglich der Realisierung des Systems stellen auch die automatische Generierung von Protokollsoftware aus der formalen Spezifikation des Protokolls und die anschließende quantitative Betrachtung der erreichten Leistungsmerkmale interessante Ergänzungen zu der vorliegenden Arbeit dar.

Literaturverzeichnis

1. Weule, H.: „Information als Produktionsfaktor". In: Görke, W.; Rinisland. H.; Syrbe, M. (Hrsg.): *Information als Produktionsfaktor. (GI Jahrestagung Karlsruhe 1992).* Berlin u.a: Springer Verlag, 1992, (Reihe Informatik aktuell), S. 3 - 19
2. Pfeiffer, T.: *Qualitätsmanagement.* München, Wien: Carl Hanser Verlag, 1993
3. Schneider, H.-J.: „INTERKAMA 92: Sensorsysteme und die Kommunikation im Feld". *atp - Automatisierungstechnische Praxis* 35 (1993) 2, S. 71 - 83
4. Rathje, J.: „Braucht die chemische Industrie den Feldbus?". *atp - Automatisierungstechnische Praxis* 36 (1994) 4, S. 22 - 30
5. Rembold, U.; Levi, P.: *Realzeitsysteme zur Prozeßautomatisierng.* München, Wien: Carl Hanser Verlag, 1994
6. Feldmann, K.; Solvie, M.: „OLCHFA: Zeitkritische, drahtlose Feldbuskommunikation". In: *Tagungsband zur inet'94 (Hamburg 1994).* Hagenburg: Network GmbH, 1994, S. 243 - 249
7. Feldmann, K.; Solvie, M.: „Einsatz von Feldbussystemen zur Übertragung von Multimedia-Daten aus dem Prozeß". In: *Tagungsband zur inet'95 (Karlsruhe 1995).* Hagenburg: Network GmbH, 1995, S. 51-57
8. ISO: *Information Technology - Open Systems Interconnection - Basic Reference Model.* IS 7498. Genf: International Organization for Standardization, 1988
9. Rose, M.T.: *The open book.* Englewood Cliffs (NJ): Prentice Hall, 1990
10. Fandel, G.; Dyckhoff, H.; Reese, J.: *Industrielle Produktionsentwicklung.* 2. Auflage. Berlin u.a.: Springer Verlag, 1994
11. Lauber, R.: *Prozeßautomatisierung - Band 1: Automatisierungsstrukturen, Prozeßrechensysteme, Echtzeit-Programmierung, Zuverlässigkeits- und Sicherheitstechnik.* 2. Auflage. Berlin u.a.: Springer-Verlag, 1989
12. Enslow, P.H.: „What is a 'Distributed' data processing system?" *IEEE Computer* 11 (1978) 1, S. 13 - 21
13. Herrtwich, R. G.; Hommel, G.: *Kooperation und Konkurrenz, Nebenläufige, verteilte und echtzeitabhängige Programmsysteme.* Berlin u.a.: Springer Verlag, 1989
14. Dietsch, H.: „Physische Eigenschaften verteilter Rechensysteme: Beiwerk oder Basis?". In: Wedekind, H. (Hrsg.): *Verteilte Systeme.* Mannheim, Wien, Zürich: BI-Wissenschaftsverlag, 1994, S. 3 - 16.
15. Rzehak, H.: „Echtzeitkommunikationssysteme". In: Encarnação, J. (Hrsg.): *Telekommunikation und multimediale Anwendungen der Informatik.* Berlin u.a.: Springer-Verlag, 1991, (Reihe Informatik Fachberichte 293), S. 631 - 641
16. Stieger, K.: „Randbedingungen für Protokolle zur transaktionsorientierten Datenvearbeitung in verteilten Realzeitsystemen". In: Encarnação, J. (Hrsg.): *Telekommunikation und multimediale Anwendungen der Informatik.* Berlin u.a.: Springer-Verlag, 1991, (Reihe Informatik Fachberichte 293), S. 643 - 656.

17. Rodd, M.H.; Zhao, G. F.; Izikowitz, I.: „RTMMS - An OSI-Based Real-Time Messaging System". *The Jounal of Real-Time Systems* 2 (1990), S. 213 - 234

18. Zhao, G. F.: *A Real-Time Messaging System for Distributed Computer Control Systems.* Ph.D.-Thesis. University of Wales, Swansea, 1990

19. Powell, D.: *Delta-4: A Generic Architecture for Dependable Distributed Computing.* Research Report ESPRIT Project 818/2252, Vol. 1. Berlin u.a.: Springer-Verlag, 1991

20. Steusloff, H.: „Funktionsstruktur, Kommunkationsstruktur und Kommunikationsmittel in Automatisierungs- und Leitsystemen". *atp - Automatisierungstechnische Praxis* 31 (1989) 5, S. 209 - 217

21. Wagner, S.: „Informationstechnische Aspekte von CIM". *atp - Automatisierungstechnische Praxis* 32 (1990) 1, S. 7 - 16

22. Ahrens, K.; Götz, E.; Möbus, J.: „Produktionsleittechnik: Divergierende Begriffe in Verfahrenstechnik und Fertigungstechnik". *atp - Automatisierungstechnische Praxis* 32 (1990) 10, S. 495 - 499

23. Adam, W.; Kinnemann, H.; Menevidis, Z.: „Beispiele einer auf Standards basierenden Kommunikationsinfrastruktur für CIM". *ZwF CIM* 87 (1992) 7, S. 358 - 361

24. Solvie, M.: „Industrielle Datenkommunikation mit Bussystemen". *DATACOM* 10 (1993) 12, S. 130 - 135 und 11 (1994) 2, S. 154 - 160

25. Kauffels, F.-J.: *Rechnernetzwerk-Systemarchitekturen und Datenkommunikation.* 3. überarbeitete und erweiterte Auflage. Mannheim, Wien, Zürich: BI-Wissenschaftsverlag, 1991

26. ICSI: *Transmission Control Protocol.* RFC 793. Marina del Rey: ICSI, 1981

27. N.N.: „Die Internet-Familie verliert an Boden". *Computer Zeitung* 26 (1995) 16, S. 16

28. Solvie, M.; Solvie, G.: „Durchgängige Kommunikation für heterogene Systeme in der rechnerintegrierten Fertigung". In: Görke, W.; Rinisland. H.; Syrbe, M. (Hrsg.): *Information als Produktionsfaktor. (GI Jahrestagung Karlsruhe 1992).* Berlin u.a: Springer Verlag, 1992, (Reihe Informatik aktuell), S. 396 - 405

29. Feldmann, K.; Franke, J.; Solvie, M.: „Integration heterogener Kommunikationsprotokolle in der automatisierten Fertigung durch Migration in MMS". *Elektronik* (1992) 2, S. 66 - 76

30. Solvie, M.: „Migration in der rechnerintegrierten Fertigung". *DATACOM* 9 (1992) 3, S. 122 - 130 und 9 (1992) 4, S. 116 - 124

31. Schneider, H.-J.: „ACHEMA 94: Sensorsysteme und die Kommunikation im Feld". *atp - Automatisierungstechnische Praxis* 36 (1994) 9, S. 10 - 31

32. Bender, K.: *PROFIBUS - DerFeldbus für die Automation.* 2. überarbeitete Auflage. München, Wien: Carl Hanser Verlag, 1992

33. Li, Y.: *Bewertung der Echtzeitfähigkeit von Feldbussystemen.* Düsseldorf: VDI-Verlag, 1993 (Fortschrittsberichte VDI Reihe 10 Nr. 235)

34. European MAP Users Group: *Manufacturing Automation Protocol Specification.* Volume I - V. Cranfield: 1989

35. ISO: *Manufacturing Message Specification.* Part I: Service Definition; Part II: Protocol Specification. IS 9506. Genf: International Organization for Standardization, 1988

36. International Robotics and Factory Automation Center: *FAIS 2.0 / Chapter 3 (MMS),* 1992

37. Stein, W.: „Objektorientierte Analysemethoden – ein Vergleich". *Informatik Spektrum* 16 (1993) 6, S. 317 – 332

38. Mühlhäuser, M.; Schill, A.: *Software Engineering für verteilte Anwendungen.* Berlin u.a.: Springer Verlag, 1992

39. Bluman, W.; Horstmann, A. (Hrsg.): *Fertigungsautomatisierung mit MMS.* Düsseldorf: VDI-Verlag, 1993

40. Rose, M.: *Prozeßautomatisierung mit dem DIN-Meßbus und dem Interbus-S.* Heidelberg: Hüthig Buch Verlag, 1993

41. Hils, F.; Lindner, K.-P.: „PROFIBUS – Der Feldbus für die Verfahrenstechnik wird erwachsen". *atp - Automatisierungstechnische Praxis* 34 (1992) 12, S. 661 – 667

42. Bonfig, K.W. u.a.: *Feldbus-Systeme.* Ehningen: Expert-Verlag, 1992

43. Borst, W.: *Der Feldbus in der Maschinen- und Anlagentechnik.* München: Franzis'-Verlag, 1992

44. Etschberger, K.: *CAN Controller area network.* München, Wien: Carl Hanser Verlag, 1994

45. Lindner, W.: „ABUS als betriebssicheres Kommunikationsnetz für Kraftfahrzeuge". In: *Tagungsband zum VDI-Seminar „Kommunikationssysteme für den Prozess- und Feldbereich" (Aachen 1991).* Düsseldorf: VDI-Bildungswerk, 1991

46. Solvie, M.: „Digitale Schnittstelle für Antriebssysteme" *pa - Produktionsautomatisierung.* 2 (1993) 4, S. 30

47. Kriesel, W.; Madelung, O. (Hrsg.): *ASI - Das Aktuator-Sensor-Interface für die Automation.* München, Wien: Carl Hanser Verlag, 1994

48. Pfeifer, T.; Heiler, K.U.: „Ziele und Anwendungen von Feldbussystemen". *atp - Automatisierungstechnische Praxis* 29 (1987) 12, S. 549 – 557

49. FICIM Project (Fraunhofer IITB): *User Requirements Study.* Deliverable 23 des ESPRIT-Projektes 5206 „FICIM". Stand: 2.11.1992

50. VDMA: *Feldbusse im Maschinen- und Anlagenbau.* Ergebnisse der VDMA-Anwenderumfrage. Frankfurt am Main: Maschinenbau Verlag GmbH, 1991

51. Trille, M.: „Dynamischer Lastausgleich in verteilten Echtzeitsystemen". *atp - Automatisierungstechnische Praxis* 35 (1993) 9, S. 527 – 532

52. Halang, W. A.; Hommel, G.; Lauber, R.: „Perspektive der Informatik in der Echtzeitverarbeitung". *Informatik Spektrum* 16 (1993) 6, S. 357 – 360

53. Stankovic, J. A.: „Misconceptions about Real-Time Computing – A serious problem for next-generation systems". *IEEE computer* (1988) 10, S. 10 – 19

54. DIN: *Informationsverarbeitung.* DIN 44300, Berlin: Beuth Verlag, 1988

55. Herrtwich, R.G.: „Echtzeit". *Informatik-Spektrum* 12 (1989) 3, S. 93 - 96

56. Spiess, P.P.: „Zur Zeitabhängigkeit von Informationswerten in verteilten Systemen". *Informatik Forschung und Entwicklung* (1987) 2., S. 131 - 146

57. Poledna, S.: „Reliability of event-triggered task activation for hard real-time systems". In: *Proceedings of the IEEE Real-Time System Symposium* (Raleigh-Durham 1993), S. 208 - 210

58. Steinmetz, R.; Herrtwich, R.G.: „Integrierte verteilte Multimedia-Systeme". *Informatik Spektrum* 14 (1991) 5, S. 249 - 260

59. Solvie, M.: „Integration of Multimedia Services and Objects into common industrial fieldbus networks". In: *Proceedings of the 10th International Conference on Advanced Science and Technology.* (ICAST Naperville (USA, IL) 1994), S. 37 - 44

60. Steinmetz, R.: *Multimedia-Technologie: Einführung und Grundlagen.* Berlin u.a.: Springer-Verlag, 1993

61. Rakow. T.C.; Löhr, M.; Moser, F.; Neuhold, E.J.; Süllow, K.: „Einsatz von Datenbanksystemen für Multimedia-Anwendungen". *it+ti - Informationstechnik und technische Informatik* 35 (1993) 3, S. 4 - 17

62. Lewe, H.: „Groupware". *Informatik Spektrum* 14 (1991) 6, S. 345 - 348

63. Grampp, K.: *Rechnerunterstützung bei Test und Schulung von Steuerungssoftware bei SMD-Bestücklinien.* München, Wien: Carl Hanser Verlag, 1995 (Dissertation)

64. Springer, G.: *Simulationsgestützte Mitarbeiterausbildung am Beispiel der Fertigungssteuerung.* Düsseldorf: VDI-Verlag, 1992 (Fortschrittsberichte VDI Reihe 2 Nr. 269)

65. Dehlinger, H.: „Learning by doing". *Business Computing* (1993) 12. S. 26 - 27

66. Feuerstake, H. J.: „Multimediale integrierte Informationsverarbeitung". *CIM Management* 12 (1995) 2, S. 50 - 55

67. Rembold, U.; Levi, P.: *Realzeitsysteme zur Prozeßautomatisierung.* München, Wien: Carl Hanser Verlag, 1994

68. Charwat, H.J.: „Prozeßführung mit Bildschirmen". In: *Einsatz intelligenter Feldgeräte in der Verfahrenstechnik.* Düsseldorf: VDI-Verlag, 1994 (Reihe VDI-Berichte, Band 1144), S. 65 - 79

69. Denzer, R.: „Optimierung der Prozeßführung durch prozeßadaptive graphische Benutzeroperflächen". In: Görke, W.; Rinisland. H.; Syrbe, M. (Hrsg.): *Information als Produktionsfaktor. (GI Jahrestagung Karlsruhe 1992).* Berlin u.a: Springer Verlag, 1992, (Reihe Informatik aktuell), S. 296 - 304

70. Kurbel, K.: „Multimedia-Unterstützung für die Fertigungssteuerung". *ZwF CIM* 87 (1992) 12, S. 664 - 668

71. Zinser, K.: „Neue Formen und Medien der Prozeßvisualisierung". *atp - Automatisierungstechnische Praxis* 35 (1993) 9, S. 499 - 504

72. Adam, W.; Fredrich, H.; Linnemann, H.: „Modellbasiertes Multimedia-Ferndiagnosesystem". *ZwF CIM* 87 (1992) 12, S. 659 - 663

73. Feldmann, K.; Sturm, J.: „Röntgenographische Analyse von Fine-Pitch-Lötstellen". *vte* (1993) 4, S. 158 - 163

74. Föhr, R.: *Photogrammetrische Erfassung räumlicher Informationen aus Videobildern.* Braunschweig: Vieweg Verlag, 1990 (Reihe Fortschritte der Robotik, Band 7)

75. Speidel, T.: „Magische Augen - Ereignisgesteuerte digitale Echtzeit-Bilderfassung". In: *industrie-elektrik + elektronik.* 40 (1995) 2, S. 24 - 25

76. N.N.: „... und sie sprechen doch!?". *Informatik Magazin* (1993) 6, S. 8 - 10

77. Fellbaum, K.-R.: „Elektronische Sprachverarbeitung". *CIM Management* 12 (1995) 2, S. 43 - 49

78. Löffler, T.: „Ein neuer Ansatz »Einparken nach Gehör« - Akustische Montageüberwachung steigert Qualität und Verfügbarkeit." *Produktion* (1994) 21, S. 3

79. Heldmann, K.: „Die Maschine muß hören lernen". *QZ* 39 (1994) 6 , S. 656 - 657

80. Hüggenberg, J.: „ Bilderrausch und Spährenklänge - Video- und Audiokompressionsverfahren im Überblick". *Elektronik* (1994) 22, S. 152 - 158

81. Meister, P.: *Multimedia Anwendungen.* Poing: Franzis'-Verlag, 1994

82. ISO/IEC.: *Information Technology - Digital Compression of Continous-Tone still images.* IS 10918, Genf: International Organization for Standardization, 1992

83. Wallace, G.K.: „The JPEG Still Picture Compression Standard". *Communications of the ACM* 34 (1991) 4, S. 30 - 44

84. Meyer, B.: „Ein Chip für alle Fälle". *Elektronik* (1994) 22, S. 164 - 170

85. ITU: *Narrow-Band Visual Telephone Systems and Terminal Equipment.* Recommendation H.320. Genf: ITU-T, 1990

86. ITU: *Video CODEC for audiovisual services at px64 kbits/s.* Recommendation H.261. Genf: ITU-T, 1990

87. Delgrossi, L.; Halstrick, C.; Hehmann, D.; Herrtwich, R.G.; Krone, O.; Sandvoss, J.; Vogt, C.: *Media Scaling for Audiovisual Communication with the Heidelberg Transport System.* IBM Technical Report 43.9305. Heidelberg: IBM European Networking Center, 1993

88. Küsters, H.: „MPEG und die Welt der Bilder". *Design & Elektronik* (1994) 5, S. 56–58

89. Weber, R.: „Von Electronic-Mail zu multimedialer Post". *Informatik-Spektrum* 17 (1994) 4, S. 222 - 231

90. Miloucheva, I.; Rebensburg, K.: „Datenautobahn für die rechnerintegrierte Produktion". *CIM Management* 12 (1995) 2, S. 36 - 42

91. Händel, R.; Huber, M. N.; Schröder, S.: *ATM networks.* 2. Auflage. Workingham u.a.: Addison-Wesley, 1994

92. Levergood, T. M.; Payne, A. C.; Gettys, J.; Treese, G. W.; Stewart, L. C.: *AudioFile: A Network-Transparent System for Distributed Audio Applications.* Technical Report CRL 93/8. DEC Cambridge Laboratories, 1993

93. Ferrari, D.; Banerjea, A.; Zhang, H.: *Network Support for Multimedia - A discussion of the Tenet Approach.* Technical Report TR-92-072 des International Computer Science Institute. Berkeley, CA: November 1992 (verfügbar via `ftp tenet.berkeley.edu` im Verzeichnis `/pub/tenet/Papers/` als Datei `FeBaZh92.ps`)

94. Adam, J.F.; Houh, H.H.; Ismert, M.; Tennenhouse, D.L.: „A network Architecture for Distributed Multimedia Systems". In: *Proceedings of the International Conference on Multimedia Computing and Systems (Boston 1994)*, S. 76 - 86

95. OLCHFA-Project (FAPS): *Evaluation of time-critical requirements.* Projektbereicht FAP-001RS/2.1/C. Erlangen: 1992

96. Pflügl, M.; Damm, A.: „Kommunikationsmechanismen verteilter Echtzeitsysteme und ihre Echtzeitfähigkeit". *Informatik Spektrum* 12 (1989) 3, S. 121 - 132

97. Föllinger, O.: *Lineare Abtastsysteme.* 3. Auflage. München: R. Oldenbourg Verlag, 1986

98. IEC: *Fieldbus Application Layer Specification.* Working Document of the IEC TC 65 WG 6, Draft 2.1, 1992

99. Pleinevaux, P.: „An improved Hard Real-Time Scheduling for the IEEE 802.5". *The Journal of Real-Time Systems* 4 (1992) 2, S. 99 - 112

100. Lamport, L.: „Time, Clock and the ordering of Events in a distributed system". *Communications of the ACM* 21 (1978) 7, S. 558 - 565

101. Hofmann, R.: „Kausalität und Zeit in parallelen und verteilten Systemen". Wedekind, H. (Hrsg.): *Verteilte Systeme.* Mannheim, Wien, Zürich: BI-Wissenschaftsverlag, 1994, S. 79 - 106

102. Hofmann, R.: *Gesicherte Zeitbezüge für die Leistungsanalyse in parallelen und verteilten Systemen.* Arbeitsberichte des IMMD der Friedrich-Alexander-Universität Erlangen-Nürnberg. Band 26. Erlangen, 1993 (Dissertation)

103. Kopetz, H.; Ochsenreiter, W.: „Clock synchronization in distributed real-time systems". *IEEE transactions on computers* 36 (1987) 8, S. 933 - 940

104. Türke, C.: „Räumlich verteilte Regelung über INTERBUS-S". In: *Tagungsband zur inet'94 (Hamburg 1994).* Hagenburg: Network GmbH, 1994, S. 150 - 157

105. OLCHFA-Project (i^2it): *Functional Specification for the British Steel Pilot Application.* Projektbereicht IIT-0020FS/4.4/C. Swansea: 1994

106. ISP-Foundation: *InterOperable Systems Project - Fieldbus Specification Revision 3.0,* 1993

107. ANSI/NEMA: *User requirements for systems supporting time-critical communications.* Final draft version of the TCCA Technical Report 12178, 1992

108. Williams, N.S.; Blair, G.S.: *Distributed Multimedia Application Study.* Technical report MPG-92-11. University of Lancaster, 1992. (verfügbar via `ftp ftp.comp.lancs.ac.uk` im Verzeichnis `/pub/mpg/` als Datei `MPG-92-11.ps.Z`)

109. Hoffman, E.: „Das BMFT-Verbundprojekt »Feldbus«". *atp - Automatisierungstechnische Praxis* 30 (1988) 5, S. 212 - 216

110. DIN: *PROFIBUS Norm Teil 1.* DIN 19245. Berlin: Beuth-Verlag, 1991

111. DIN: *PROFIBUS Norm Teil 2.* DIN 19245. Berlin: Beuth-Verlag, 1991

112. DIN: *PROFIBUS Dezentrale Peripherie (DP).* Gelbdruck des Normentwurfes zur DIN 19245 Teil 3, Stand Oktober 1994. Berlin: Beuth Berlag, 1994

113. Bender, K.: „PROFIBUS - Der offene Feldbusstandard". *CIM Management* 12 (1995) 2, S. 11 - 15

114. Schwörer, T.: „PROFIBUS in der Verfahrenstechnik". *Elektronik plus* (1993) 1, S. 80 - 84

115. UTE: *NF Pr C 46-603 Draft Standard - Data Link Layer.* Englische Übersetzung, Courbevoie: Union Technique de l'Electricité, 1990

116. UTE: *NF C 46-606 Couche Application Service de Messagerie.* Norme expérimentale Février 1992. Paris: Union Technique de l'Electricité, 1992

117. UTE: *NF C 46-602 Application Layer Periodic and Aperiodic Services.* Englische Übersetzung, 5.10.1990. Courbevoie: Union Technique de l'Electricité, 1990

118. Simonot, F.; Song, Y.O.; Thomesse, J.P.: „Modelling and analysis of exchange of messages in field bus FIP". *IEEE network* (1990) 9, S. 142 - 148

119. ISO: *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1).* IS 8824. Genf: International Organization for Standardization, 1987

120. He, J.; Mammeri, Z.; Thomesse, J. P.: „Clock synchronisation in real-time distributed system based on FIP field bus". *IEEE network* (1990) 9, S. 135 - 141

121. Izikowitz, I.; Solvie, M.: „The OLCHFA-project: industrial needs for wireless communication - Wireless data transmission and application layer protocol support for time-critical communication". In: P.P. Spiess (Hrsg.) *Euro-Arch'93, Europäischer Informatik Kongreß „Architektur von Rechensystemen" (München 1993).* Berlin u.a.: Springer Verlag, 1993, S. 358 - 377 (Reihe Informatik aktuell)

122. Feldmann, K.; Solvie, M.: „Die Uhr im Feldbus". *Elektronik* (1995) 6, S. 112 - 122

123. OLCHFA-Project (FAPS): *Specification of time-critical fieldbus application layer extensions.* Projektbereicht FAP-003FS/2.2/R. Erlangen: 1993

124. Club FIP: *FIP Toolbox 3 Reference Manual.* Version 3, Ausgabestand 12.08.1992. Nancy: Centre de Compétence FIP/Club FIP, 1992

125. Club FIP: *FIP Toolbox version 3.1 Addendum to the user manual.* Stand 10.05.1993. Nancy: Centre de Compétence FIP/Club FIP, 1993

126. Patz, M.: „Starthilfe für PROFIBUS-Anwender". *Elektronik plus* (1993) 1, S. 46 - 51

127. Klinker, W.: „Graphisch interaktive Konfigurationssoftware für Datennetze". *atp - Automatisierungstechnische Praxis* 35 (1993) 8, S. 476 - 480

128. Solvie, M.: „Configuration of a Distributed Time-Critical Fieldbus System". In: *Proceedings of the second international workshop on configurable distributed systems* (Pittsburg (PA), 1994) Los Alamitos: IEEE computer society, 1994, S. 211

129. Martin, J.; Odell, J.: *Object oriented analysis and design.* Englewood Cliffs: Prentice-Hall, 1992

130. OLCHFA-Project (FAPS): *EXF - Extended FIP Configuration Language Specification Document.* Projektbereicht FAP-004FS/2.3/R. Erlangen: 1994

131. OLCHFA-Project (FAPS): *Users guide to OLCHFA Configuration tools.* Projektbereicht FAP-006FS/2.5/P. Erlangen: 1994

132. Solvie, M.: „Feldbusse in der Anwendung: Der OLCHFA-Feldbus bei zwei Pilotanwendungen". *pa - Produktionsautomatisierung* 3 (1994) 5/6. S. 35 - 38

133. IEEE: *Distributed Queue Dual Bus (DQDB) Subnetwork of a Metropolitan Area Network (MAN).* IEEE Standard 802.6-1990. New York: IEEE, Inc., 1990

134. CENELEC: *Fieldbus standard for industrial control systems - Part 2: Physical layer specification and service definition.* Europäische Norm EN 61158-2. Brüssel: CENELEC, 1994

135. Tanenbaum, A.S.: *Computer networks.* Englewood Cliffs: Prentice-Hall, 1989

136. Turletti, T.; Huitema, C.: *Packetization of H.261 video streams.* Internet Engeneering Task Force Audio-Video Transport Working Group. Internet-Draft. INRIA, 1994

137. ISO: *Information Processing Systems - Open Systems Interconnection - Protocol Specification for the Associtation Control Service Element.* ISO. Revised Final Text of DIS 8650. Genf: International Organization for Standardization, 1988

138. Booch, G.: *Object oriented design with applications.* Redwood City: Benjamin/Cummings Publishing Company, 1991

139. Müller, H.: *Diskrete algebraische Strukturen.* Arbeitsberichte des IMMD der Friedrich-Alexander-Universität Erlangen-Nürnberg. Band 20. Erlangen, 1987

140. Hofmann, F.: *Betriebssysteme: Grundkonzepte und Modellvorstellungen.* Stuttgart: B.G. Teubner, 1984

141. Deitel, H.M.: *An introduction to Operating Systems.* Revised first edition. Reading (MA) u.a.: Addisson-Wesley Publishing Company, 1984

142. Keppke, A.: „Just in Time". *Elektronik* (1992) 8, S. 52 - 134 und (1992) 9, S. 202 - 210

143. Raab, H.: *Handbuch Industrieroboter.* Braunschweig, Wiesbaden: Vieweg, 1981

144. Spur, G., Schöer, K.: „Kinematische Modellierung und numerische Parameteridentifikation bei Industruierobotern". In: Pritschow, G., Spur, G., Weck, M. (Hrsg.): *Roboteranwendungen in der flexiblen Fertigung.* München, Wien: Carl Hanser Verlag, 1994, S. 83 - 106

145. Ránky, P.G.; Ho, C.Y.: *Robot modelling.* Berlin u.a.: Springer-Verlag, 1985

146. Wloka, K.: *Robotersysteme.* Band 1: Technische Grundlagen. Berlin. u.a.: Springer-Verlag, 1992

147. Raja, P.; Gonzalo, U.: „A simple encoder for fieldbus applications". *ACM Computer Communications Review* 23 (1993) 1, S. 34 - 44

148. ISO: *Information processing systems; Open Systems Interconnection; Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1).* IS 8825. Genf: International Organization for Standardization, 1987

149. Bärnreuther, B.: *Ein Beitrag zur Bewertung des Kommunikationsverhaltens von Automatisierungsgeräten in flexiblen Produktionszellen.* München, Wien: Carl Hanser Verlag, 1992 (Dissertation)

150. ITU: *Information technology - Open Systems Interconnection - Systems Management Overview.* Recommendation X.701 der ITU-T (entspricht ISO/IEC 10040). Genf: ITU-T, 1992

151. ITU: *Management Framework for Open Systems Interconnection (OSI) for CCITT Applications.* Recommendation X.700 der ITU-T (entspricht ISO/IEC 7498-4). Genf: ITU, 1992

152. Dünkel, C.; Paradies, T.: „Übertragung komprimierter isochroner Datenströme". In: B. Wolfinger (Hrsg.): *Innovationen bei Rechen- und Kommunikationssystemen.* Berlin u.a.: Springer Verlag, 1994, (Reihe Informatik aktuell), S. 188 – 195

153. ISO: *USA contribution on the Time Management Function.* Arbeitspapier N 79 der ISO TC 184 SC5 WG 2, 1992

154. Pimentel, J.R.: *Communication networks for manufacturing.* Englewood Cliffs: Prentice Hall, 1990

155. Spaniol, O.: „High speed LANs". In: *Tutorium „Kommunikation in verteilten Systemen" (Stuttgart 1989).* Deutsche Informatik Akademie, 1989, S. 4-5 – 4-9

156. Vogt, R.; Bender, K.: „Erhöhung der Zuverlässigkeit busorientierter Automatisierungssysteme". *atp - Automatisierungstechnische Praxis* 29 (1987) 2, S. 53 – 57

157. Hogrefe, D.: *Estelle, LOTOS und SDL.* Berlin u.a.: Springer Verlag, 1989

158. ITU: *Information Processing Systems - Open Systems Interconnection - Basic Reference Model: Conventions for the definition of OSI services.* Recommendation X.210 (entspricht ISO/IEC 10731). Genf: ITU, 1993

159. ITU: *Message Sequence Chart.* Recommendation Z.120. Genf: ITU-T, 1993

Anhang A Darstellungskonventionen

Zur Darstellung von Diensten und der zeitlichen Abfolge des Austausches von Nachrichten wird in der gesamten Arbeit auf die diesbezüglichen Vorgaben aus den Standards ITU-T X.210 [158] für die Darstellung der Dienste und ITU-T Z.120 [159] für die Darstellung der Reihenfolge des Austausches von Nachrichten zurückgegriffen, die nachfolgend kurz eingeführt werden.

A.1 Definition von Diensten

Grundsätzlich wird zwischen *bestätigten* und *unbestätigten* Diensten unterschieden. Bestätigte Dienste beinhalten eine Dienstanforderung und eine dazugehörige Antwort des korrespondierenden Dienstbenutzers. Unbestätigte Dienste beinhalten nur die Zustellung der Dienstanforderung. Der Initiator eines Dienstes, der auch als *Requester* bezeichnet wird, fordert dabei dessen Ausführung durch Absetzen des Dienstelementes der „Anforderung" (*Request, req.*) an. Die Anforderung wird über das Kommunikationssystem zu dem korrespondierenden Dienstbenutzer, der auch als *Responder* bezeichnet wird, gesendet. Dieser erhält die Dienstanforderung mit dem Dienstelement der „Ankündigung" (*Indication, ind.*) und sendet bei bestätigten Diensten eine „Antwort" (*Response, res.*), die beim Requester als „Bestätigung" (*Confirmation, cnf.*) entgegengenommen wird.

Zur Darstellung der Parameter der Dienstelemente wird auf die allgemein bekannte tabellarische Form zurückgegriffen, wie sie z.B. auch in den FMS- und MMS-Standards verwendet wird. Die verwendeten Abkürzungen haben dabei folgende Bedeutung [158]:

„.req"	Dienstanforderung	„.ind"	Dienstankündigung
„.res"	Dienstantwort	„.cnf"	Dienstbestätigung
„M"	Zwingend vorgeschriebener Dienstparameter („Mandatory")		
„U"	Optionaler Parameter („User option")		
„S"	Parameter ist eine Auswahl unter zwei oder mehreren („Selection")		
„C"	Parameter wird in Abhängigkeit des Wertes eines anderen Parameters benötigt oder nicht („Conditional")		

Ein nachgestelltes „(=)" bei den Parametern der Dienstelemente der Ankündigung und der Bestätigung zeigt die semantische Gleichwertigkeit der Werte der Parameter mit denen der zugehörigen Dienstanforderung bzw. Dienstantwort an.

In dem dargestellten Beispiel (vgl. Tabelle A.1) beinhalten die Dienstanforderung und -ankündigung grundsätzlich den Parameter „Parameter_1" und dann entweder den Parameter „Parameter_2" oder „Parameter_4". In ersterem Fall ist auch die Verwendung des Parameters „Parameter_3" vorgeschrieben. Die Angabe des Parameters „Parameter_5" obliegt der Wahl des Benutzers. Die Dienstantwort und -bestätigung sind parameterlos.

Einige Dienste in den vorgestellten Systemen beruhen auf lokalen Protokollaktionen und zum Teil auf ergänzenden Aktivitäten der darunterliegenden Protokollinstanzen, beinhalten aber keine direkte Interaktion mit der Partnerinstanz des Protokolls. Bei der Beschreibung dieser Dienste werden trotzdem die gleichen Begriffe verwendet.

Parametername	req	ind	res	cnf
Argument	**M**	**M(=)**		
Parameter_1	M	M(=)		
Parameter_2	S	S(=)		
Parameter_3	M	M(=)		
Parameter_4	S	S(=)		
Parameter_5	U	U(=)		
Result(+)			**S**	**S(=)**
Result(-)			**S**	**S(=)**

Tabelle A.1: Parameter eines beispielhaften Dienstes

A.2 Darstellung des Nachrichtenaustausches mit *Message Sequence Charts*

Message Sequence Charts (MSC) dienen der Darstellung der zeitlichen Abfolge des Austausches von Nachrichten zwischen Bestandteilen des Systems, die als „Instanzen" bezeichnet werden, und deren Umgebung. Die wesentlichen graphischen Elemente, die daraus in dieser Arbeit benutzt werden, und deren Bedeutungen sind in Bild A.1 zusammengefaßt. Die Zeit verläuft in einem MSC-Diagramm entlang einer jeden Instanzachse von oben nach unten. Eine globale Zeitreferenz existiert nicht. Die Ordnung von Ereignissen in unterschiedlichen Instanzen erfolgt nur durch die ausgetauschten Nachrichten, die gesendet werden, bevor sie empfangen werden [159].

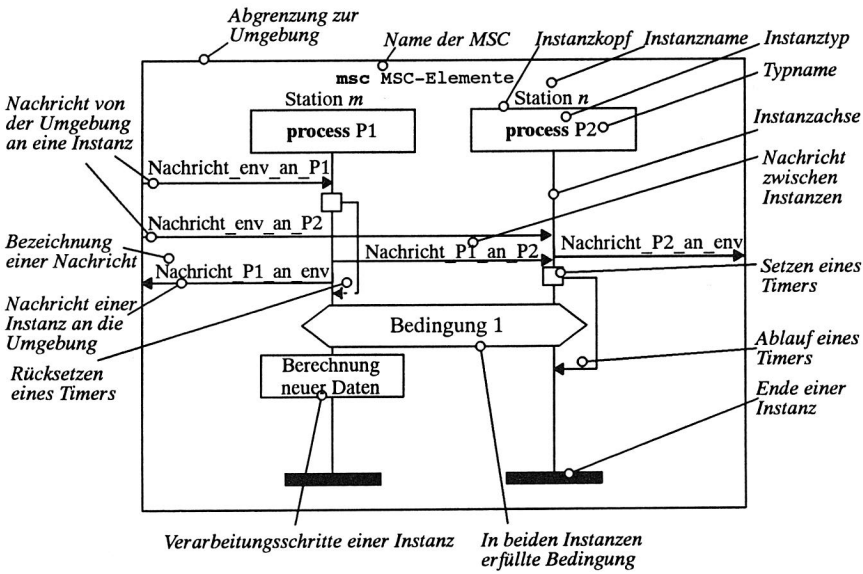


Bild A.1: Benutzte graphische Elemente von MSC

Anhang B Ergänzende Darstellungen zur Sicherungsschicht

B.1 Formate der Protokolldateneinheiten

Der Austausch von Daten erfolgt für alle Protokolle unter Nutzung definierter Protokolldateneinheiten. Die benötigten Formate für die Protokolle der Sicherungsschicht sind nachfolgend dargestellt.

B.1.1 DQDFB

Eine beliebige DQDFB-Protokolldateneinheit hat die in Bild B.1 dargestellte Struktur.

ACF (1 Byte)	Segment (35 Byte)

(a) Rahmenformat

BUSY (1 Bit)	SL_TYPE (2 Bit)	REQUEST (2 Bit)	reserved (3 Bit)

(b) Access Control Field (ACF)

SL_TYPE	Bedeutung
0 0	QA-Rahmen
0 1	PA-Rahmen
1 0	MN-Rahmen
1 1	*reserviert*

(c) Rahmentypen (SL_Type, Slottype)

REQUEST	Bedeutung
0 0	Keine Anforderung
0 1	Niederpriore Anforderung
1 0	Hochpriore Anforderung
1 1	*verboten*

(d) Bedeutung des REQUEST-Parameters

Bild B.1: Allgemeines DQDFB-Rahmenformat

Der Wert des BUSY-Feldes im ACF gibt an, ob der Rahmen frei („BUSY=0") oder belegt ist („BUSY=1"). Der Typ des Übertragungsrahmens wird durch den Wert des Parameters „SL_TYPE" festgelegt. Die Bits im Feld „REQUEST" werden zur Anforderung von aperiodischen Übertragungsrahmen genutzt.

QA-Rahmen (vgl. Bild B.2) werden als Nutzdaten im Segment von DQDFB-Rahmens übertragen. Sie beinhalten 3 Byte Kontrollinformationen („QA-Segment Header") und 32 Byte Nutzdaten. Die Kontrollinformationen enthalten die Adresse der Zielstation („DA", „Destination Address"),

QA Segment Header (3 Byte)	QA Segement Payload (32 Byte)

(a) QA-Rahmenformat

DA (1 Byte)	PAYLOAD TYPE (2 Bit)	PAYLOAD PRIORITY (2 Bit)	SA (1 Byte)	HCS (4 Bit)

(b) QA-Segment Header

PAYLOAD_TYPE	Bedeutung
0 0	Empty
0 1	Publish-Request
1 0	Publish-Value
1 1	Message

(c) Kennung des Nutzdatentyps

PAYLOAD PRIORITY	Bedeutung
0 0	*reserviert*
0 1	Niederpriore Anforderung
1 0	Hochpriore Anforderung
1 1	*reserviert*

(d) Priorität der übertragenen Information

Bild B.2: Format der QA-Rahmen

die Adresse der Quellstation („SA", „Source Address"), den Typ der übertragenen Nutzdaten, die Priorität des Dienstes und eine 4 Bit lange Prüfsumme („HCS", „Header Check Sequence"). Die Anordnung der Informationen im QA- und PA-Segment Header erlaubt der Protokollinstanz von DQDFB die schnellstmögliche Evaluierung des Typs des Datenübertragungsrahmens und damit die Bestimmung der auszuführenden Aktion. Der Aufbau von PA-Rahmen (vgl. Bild B.3) ist dem von QA-Rahmen sehr ähnlich. Im Kontrollfeld wird jedoch keine Anzeige der Priorität benötigt.

PA Segment Header (3 Byte)	PA Segement Payload (32 Byte)

(a) PA-Rahmenformat

DA (1 Byte)	PAYLOAD TYPE (2 Bit)	reserved (2 Bit)	SA (1 Byte)	HCS (4 Bit)

(b) PA-Segment Header

PAYLOAD_TYPE	Bedeutung
0 0	*reserviert*
0 1	Publish-Request
1 0	Publish-Value
1 1	Message

(c) Kennung des Nutzdatentyps

Bild B.3: Format der PA-Rahmen

DQDFB stellt Mechanismen zur Segmentierung von Nutzdaten des Dienstbenutzers zur Verfügung. Die aus diesen Nutzdaten gebildete Protokolldateneinheit wird als „Initial MAC Protocol Data Unit" bezeichnet und besteht aus einer Information über die Länge der Nutzdaten und diesen selbst. Die einzelnen Segmente der IMPDU werden in aufeinanderfolgenden DMPDUs übertragen. Diese enthalten neben Verwaltungsinformationen je ein Segment der zu übertragenden Nutzdaten (vgl. Bild B.4).

Payload length (8 Bit)	IMPDU-Payload (0-247 Byte)

(a) IMPDU-Format

DMPDU Header (1 Byte)	Segment (31 Byte)

(b) DMPDU-Format

S_TYPE (2 Bit)	Segment # (6 Bit)

(c) DMPDU-Header

S_TYPE	Bedeutung
0 0	COM
0 1	BOM
1 0	EOM
1 1	SSM

(d) Segmenttypen (S_Type)

Bild B.4: Festlegungen bezüglich der IMPDU und DMPDU

B.1.2 DRLSP

Nachfolgend werden in den Bildern B.5 bis B.9 die Protokolldateneinheiten vorgestellt, die von DRSLP definiert benutzt werden. Die Bedeutung der Parameter ist selbsterklärend.

VarID (1 Byte)	Variable-value (1-245 Byte)	PAYLOAD CRC (1 Byte)

Bild B.5: Publish-Value-PDU

VarID (1 Byte)	reserviert (29 Byte)	PAYLOAD CRC (1 Byte)

Bild B.6: Publish-Request-PDU

MM_Stream_ID (1 Byte)	Length (1 Byte)	MM-data (1-244 Byte)	PAYLOAD CRC (1 Byte)

Bild B.7: PDU zum Austausch von Daten aus kontinuierlichen Multimedia-Datenströmen

Length (1 Byte)	DSAP (4 Bit)	SSAP (4 Bit)	Message-data (aperiodisch:1-235 Byte, periodisch: 1-27 Byte)	PAYLOAD CRC (1 Byte)

Bild B.8: PDU zum Austauch von unbestätigten Nachrichten

Length (1 Byte)	MCF (1 Byte)	DSAP (4 Bit)	SSAP (4 Bit)	Message-data (27 Byte)	PAYLOAD CRC (1 Byte)

(a) Rahmenaufbau

Message type (2 Bit)	Sequence number (6 Bit)

(b) Message Control Field (MCF)

PAYLOAD_TYPE	Bedeutung
0 0	*reserviert*
0 1	Data
1 0	Acknowledge
1 1	*reserviert*

(c) Kennung des Nachrichtentyps (MCF)

Bild B.9: PDU zum Austauch von bestätigten Nachrichten

B.2 Abbildung der Dienste

Tabelle B.1 stellt zusammenfassend die Reaktion der DQDFB-Protokollinstanz einer Station *n* auf das Empfangen einer Protokolldateneinheit über einen beliebigen Bus dar. Dabei repräsentiere $m \neq n$ eine einzelne andere Station, *x* eine Stationsgruppe, in der die Station *n* nicht enthalten ist, *y* eine Stationsgruppe, in der die Station *n* enthalten ist, und *z* eine beliebige Stationsgruppe. Es werden nur die wesentlichen Parameter der Protokolldateneinheiten und Dienstaufrufe dargestellt. Nach dem Empfang einer Protokolldateneinheit können das Senden einer Protokolldateneinheit und die eventuell anstehende Weitergabe von Informationen an die Instanz von DRSLP parallel ausgeführt werden.

DQDFB (n) Empfang	**DQDFB (n) Senden**	**DQDFB (n) ⧫ DRSLP (n)**
PA-PR (n, data)	QA-E	MAP-UNITDATA.ind (PR, data)
PA-PV (n, *frei*)	PA-PV (255, *bel.*, data)	MA-SENT.ind
PA-PR (m, data)	PA-PR (m, data)	
PA-PV (m, *frei*)	PA-PV (m, *frei*)	
PA-PV (255, *bel.*, data)	PA-PV (255, *bel.*, data)	MAP-UNITDATA.ind (PV, data)
PA-MSG (n, *frei*)	PA-MSG (x, *bel.*, data)	MA-SENT.ind
PA-MSG (m, *frei*)	PA-MSG (m, *frei*)	
PA-MSG (y, *bel.*, data)	PA-MSG (y, *bel.*, data)	MAP-UNITDATA.ind (MSG, data)
PA-MSG (n, *bel.*, data)	QA-E	MAP-UNITDATA.ind (MSG, data)
PA-MSG (x, *bel.*, data)	PA-MSG (x, *bel.*, data)	
QA-PR (n, data)	QA-E	MAQ-UNITDATA.ind (PR, data)
QA-PR (m, data)	QA (m, data)	
QA-PV (255, data)	QA-PV (255, data)	MAQ-UNITDATA.ind (PV, data)
QA-MSG (y, *bel.*, data)	QA-MSG (y, *bel.*, data)	MAQ-UNITDATA.ind (MSG, data)
QA-MSG (n, *bel.*, data)	QA-E	MAQ-UNITDATA.ind (MSG, data)
QA-MSG (x, *bel.*, data)	QA-MSG (x, *bel.*, data)	
QA-E *(Zugriff gestattet)*	QA-PR (m, data) *oder* QA-PV (255, data) *oder* QA-MSG (x, data) *(je nach vorliegender Anforderung)*	MA-SENT.ind
QA-E *(Zugriff verboten)*	QA-E	

Tabelle B.1: Reaktion der DQDFB-Protokollinstanz auf den Empfang von PDUs

Der Typ der Protokolldateneinheit, die nach der Entgegennahme einer korrekten Dienstanforderung von der DRSLP-Protokollinstanz gebildet und in die entsprechende Warteschlange eingereiht wird, sowie die Reaktion der DQDFB-Protokollinstanz auf das Senden der Protokolldateneinheit sind in Tabelle B.2 dargestellt. Die Reaktionen einer Instanz von DRSLP auf die Entgegennahme von Dienstankündigungen sind in Tabelle B.3 zusammenfassend dargestellt.

DRSLP ⧫ DQDFB	**DQDFB-PDU-Typ**	**DQDFB ⧫ DRSLP**
MAP-UNITDATA.req (PV, data)	PA-PV (255, *bel.*, data)	MA-SENT.ind
MAP-UNITDATA.req (x, MSG, data)	PA-MSG (x, *bel.*, data)	
MAQ-UNITDATA.req (m, PR, data)	QA-PR (m, data)	
MAQ-UNITDATA.req (PV, data)	QA-PV (255, data)	
MAQ-UNITDATA.req (x, MSG, data)	QA-MSG (x, data)	

Tabelle B.2: Reaktion der DQDFB-Protokollinstanz auf Dienstanforderungen

DQDFB ⧫ DRSLP	**DRSLP ⧫ DQDFB**	**DRSLP ⧫ Dienstbenutzer**
MAP-UNITDATA.ind (PR, data)	MAP-UNITDATA.req (PV,data)	
MAP-UNITDATA.ind (PV, data)		DL-RECEIVED.ind *bei Variablen* DL-MM-EXCHANGE.ind (data) *bei Multimedia-Datenströmen*
MAQ-UNITDATA.ind (PV, *leer*)		DL-SENT.ind
MAP-UNITDATA.ind (MSG, DSAP, data)		DL-SEND-MSG.ind *am DSAP*
MA-SENT.ind ( )		
MAQ-UNITDATA.ind (PR, data)	MAQ-UNITDATA.req (PV,data)$^{+}$	
MAQ-UNITDATA.ind (PV, data)		DL_RECEIVED.ind (data)
MAQ-UNITDATA.ind (MSG, DSAP, data)		DL_RECEIVED_MSG.ind (data) *am DSAP*

Tabelle B.3: Reaktion der Instanz von DRSLP auf die Entgegennahme von Dienstdateneinheiten.

Die Interaktionen zwischen einer Instanz vonDRSLP und deren Dienstbenutzer sind in Tabelle B.4 überblicksweise zusammengestellt. Die Abkürzungen „aper" und „per" stehen dabei für die Anforderung eines *aperiodischen* bzw. *periodischen* Nachrichtenaustausches.

Dienstbenutzer ⧫ DRSLP	**DRSLP ⧫ DQDFB** *(danach)* **DQDFB ⧫ DRSLP**	**DRSLP ⧫ Dienstbenutzer**
DL-READ-LOC.req (ID)		DL-READ-LOC.cnf
DL-WRITE-LOC.req (ID)		
DL-UPDATE.req (list)	MAQ-UNITDATA.req (PR, data)	
DL-PUBLISH.req (ID)	MAQ-UNITDATA.req (PV, data)	
DL-MM-EXCHANGE.req		
DL-SEND-MSG.req (aper., data)	MAQ-UNITDATA.req (MSG, data)	
DL-SEND-MSG.req (per, data)	MAP-UNITDATA.req (MSG, data)	

Tabelle B.4: Reaktion der DRSLP-Instanz auf Anforderungen des Dienstbenutzers

Anhang C Ergänzende Darstellungen zur Anwendungsschicht

C.1 Protokolldateneinheiten des LLI

Die vom LLI für das Senden von Nachrichten zu verwendende PDU hat den in Bild C.1 dargestellten Aufbau.

Function-code (4 bit)	CR-ID (2 bit)	reserved (2 bit)	Data (0–236 byte)

(a) Aufbau

Function code	Bedeutung	Function code	Bedeutung
0 0 0 0	Associate Request	0 1 0 1	DTC Response
0 0 0 1	Associate Response (+)	0 1 1 0	DTA Request
0 0 1 0	Associate Resonse (-)	0 1 1 1	DTA Acknowledge
0 0 1 1	Abort Request	1 0 0 0	DTU Request
0 1 0 0	DTC Request	1001 - 1111	*reserviert*

(b) Bedeutung des Funktionskodes (Function Code, FC)

Bild C.1: Aufbau der LLI–PDU

C.2 RTSIMS-Protokolldateneinheiten in ASN.1

Wie auch bei PROFIBUS-FMS, so bestehen Protokolldateneinheiten von RTSIMS aus einem Teil fester Länge und Struktur und einem variablen Teil, der nicht bei allen Protokolldateneinheiten vorhanden sein muß. Der feste Teil einer PDU beinhaltet, wie in Bild C.2 veranschaulicht, zwei Oktete.

PDU-Type (3 bit)	Service-Type (5 bit)	Invoke ID (6 bit)	Service Mode (2 bit)

Bild C.2: Aufbau des festen Bestandteils einer RTSIMS-PDU

In diesen zwei Oktetten sind der Typ der Protokolldateneinheit („PDU-Type"), der Diensttyp („Service-Type"), die Aufrufkennung („Invoke ID") und eine Kennung des Modus der Dienstausführung (hoch-/niederprior, gesicherte/ungesicherte Nachrichtenübertragung, „Service Mode") enthalten. Sowohl die Aufrufkennung als auch der Modus der Dienstausführung sind nur bei PDUs notwendig, die eine Anforderung eines bestätigten Dienstes beinhalten.

Der variable Teil einer Protokolldateneinheit kann zwischen 0 und 233 Byte Daten enthalten. Die Kodierung der Protokolldateneinheiten erfolgt gemäß der Darstellungen in Kapitel 6.3. Die Definition der Protokolldateneinheiten in ASN.1 stellt sich wie folgt dar:

```
RTSIMS DEFINITIONS ::= BEGIN
RTSIMS-PDU ::= CHOICE {  -- Kodierung in „PDU-Type"
   confirmed-RequestPDU [1]  IMPLICIT Confirmed-RequestPDU,
   confirmed-ResponsePDU[2]  IMPLICIT Confirmed-ResponsePDU,
   confirmed-ErrorPDU   [3]  IMPLICIT Confirmed-ErrorPDU,
   unconfirmedPDU       [4]  IMPLICIT Unconfirmed-PDU,
   rejectPDU            [5]  IMPLICIT Reject-PDU,
   initiatePDU          [6]  IMPLICIT Initiate-PDU
}
Confirmed-RequestPDU ::= SEQUENCE {
   invoke-id                 InvokeID, -- Kodierung in „Invoke-ID"
   confirmedServiceRequest   ConfirmedServiceRequest
}
Confirmed-ResponsePDU ::= SEQUENCE {
   invoke-id                 InvokeID, -- Kodierung in „Invoke-ID"
```

```
   confirmedServiceResponse  ConfirmedServiceResponse
}
Confirmed-ErrorPDU ::= SEQUENCE {
   invoke-id         InvokeID, -- Kodierung in „Invoke-ID"
   serviceError      ServiceError
}
Unconfirmed-PDU ::= SEQUENCE {
   invoke-id             InvokeID, -- Kodierung in „Invoke-ID", Wert beliebig
   unconfirmedService    UnconfirmedService
}
Reject-PDU ::= SEQUENCE {
   invoke-id    InvokeID, -- Kodierung in „Invoke-ID"
   reject       Reject
}
Initiate-PDU ::= SEQUENCE {
   invoke-id      InvokeID, -- Kodierung in „Invoke-ID", Wert beliebig
   initiatepdu    InitiatePDU
}
InvokeID ::= INTEGER (0..63) -- Kodierung in 6 bit

ConfirmedServiceRequest ::= CHOICE { -- Kodierung in „Service-Type"
   create                 [0]  IMPLICIT Create-Request,
   delete                 [1]  IMPLICIT Delete-Request,
   status                 [2]  IMPLICIT Status-Request,
   get-attributes         [3]  IMPLICIT Get-attributes-Request,
   set                    [4]  IMPLICIT Set-Request,
   access-control         [5]  IMPLICIT Access-control-Request,
   take-image             [6]  IMPLICIT Take-image-Request,
   subscribe-cmd          [7]  IMPLICIT Subscribe-cmd-Request,
   unsubscribe-cmd        [8]  IMPLICIT Unsubscribe-cmd-Request,
   coder-control          [9]  IMPLICIT Coder-control-Request,
   identify               [10] IMPLICIT Identify-Request,
   load-table             [11] IMPLICIT Load-table-Request,
   start                  [12] IMPLICIT Start-Request,
   stop                   [13] IMPLICIT Stop-Request,
   read                   [14] IMPLICIT Read-Request,
   rt-read                [15] IMPLICIT RT-Read-Request,
   initiatedownloadseq    [16] IMPLICIT Initiatedownloadseq-Request,
   downloadsegment        [17] IMPLICIT Downloadsegment-Request,
   terminatedownloadseq   [18] IMPLICIT Terminatedownloadseq-Request,
   initiateuploadseq      [19] IMPLICIT Initiateuploadseq-Request,
   uploadsegement         [20] IMPLICIT Uploadsegment-Request,
   terminateuploadseq     [21] IMPLICIT Terminateuploadseq-Request,
   requestdomaindownload  [22] IMPLICIT Requestdomaindownload-Request,
   requestdomainupload    [23] IMPLICIT Requestdomainupload-Request
}
ConfirmedServiceResponse ::= CHOICE { -- Kodierung in „Service-Type"
   create                 [0]  IMPLICIT Create-Response,
   delete                 [1]  IMPLICIT Delete-Response,
   status                 [2]  IMPLICIT Status-Response,
   get-attributes         [3]  IMPLICIT Get-attributes-Response,
   set                    [4]  IMPLICIT Set-Response,
   access-control         [5]  IMPLICIT Access-control-Response,
   take-image             [6]  IMPLICIT Take-image-Response,
   subscribe-cmd          [7]  IMPLICIT Subscribe-cmd-Response,
   unsubscribe-cmd        [8]  IMPLICIT Unsubscribe-cmd-Response,
   coder-control          [9]  IMPLICIT Coder-control-Response,
   identify               [10] IMPLICIT Identify-Response,
   load-table             [11] IMPLICIT Load-table-Response,
   start                  [12] IMPLICIT Start-Response,
   stop                   [13] IMPLICIT Stop-Response,
   read                   [14] IMPLICIT Read-Response,
   rt-read                [15] IMPLICIT RT-Read-Response,
   initiatedownloadseq    [16] IMPLICIT Initiatedownloadseq-Response,
   downloadsegment        [17] IMPLICIT Downloadsegment-Response,
```

```
   terminatedownloadseq [18] IMPLICIT Terminatedownloadseq-Response,
   initiateuploadseq    [19] IMPLICIT Initiateuploadseq-Response,
   uploadsegment        [20] IMPLICIT Uploadsegment-Response,
   terminateuploadseq   [21] IMPLICIT Terminateuploadseq-Response,
   requestdomaindownload[22] IMPLICIT Requestdomaindownload-Response,
   requestdomainupload  [23] IMPLICIT Requestdomainupload-Response
}
ServiceError ::= CHOICE {  -- Kodierung in „Service-Type"
   create               [0]  IMPLICIT Error-Type,
   delete               [1]  IMPLICIT Error-Type,
   status               [2]  IMPLICIT Error-Type,
   get-attributes       [3]  IMPLICIT Error-Type,
   set                  [4]  IMPLICIT Error-Type,
   access-control       [5]  IMPLICIT Error-Type,
   take-image           [6]  IMPLICIT Error-Type,
   subscribe-cmd        [7]  IMPLICIT Error-Type,
   unsubscribe-cmd      [8]  IMPLICIT Error-Type,
   coder-control        [9]  IMPLICIT Error-Type,
   identify             [10] IMPLICIT Error-Type,
   load-table           [11] IMPLICIT Error-Type,
   start                [12] IMPLICIT Error-Type,
   stop                 [13] IMPLICIT Error-Type,
   read                 [14] IMPLICIT Error-Type,
   rt-read              [15] IMPLICIT Error-Type,
   initiatedownloadseq  [16] IMPLICIT Error-Type,
   downloadsegment      [17] IMPLICIT Error-Type,
   terminatedownloadseq [18] IMPLICIT Error-Type,
   initiateuploadseq    [19] IMPLICIT Error-Type,
   uploadseq            [20] IMPLICIT Error-Type,
   terminateuploadseq   [21] IMPLICIT Error-Type,
   requestdomaindownload[22] IMPLICIT Error-Type,
   requestdomainupload  [23] IMPLICIT Error-Type
}
Error-Type ::= SEQUENCE {
   error-class              [0]  IMPLICIT Error-Class,
   additional-code          [1]  IMPLICIT INTEGER OPTIONAL,
   additional-description   [2]  IMPLICIT VisibleString OPTIONAL
}
Error-Class ::= SEQUENCE {
   CHOICE {
      rtvfd-state           [0]  IMPLICIT Unsigned8  {
         other    (0)
      },
      application-reference[1]  IMPLICIT Unsigned8  {
         other                    (0),
         application-unreachable  (1)
      },
      definition            [2]  IMPLICIT Unsigned8  {
         other                    (0),
         object-undefined         (1),
         object-attribute-error   (2),
         name-already-exists      (3)
      },
      resource           [3]  IMPLICIT Unsigned8  {
         other              (0),
         memory-unavailable (1)
      },
      service            [4] IMPLICIT Unsigned8  {
         other                          (0),
         object-state-conflict          (1),
         pdu-size                       (2),
         parameter-error                (3),
         variable-produced-remotely     (4)
      },
      access             [5] IMPLICIT Unsigned8  {
         other                    (0),
```

```
        access-denied              (1),
        object-attribute-error     (2),
        object-non-existing        (3),
        type-conflict              (4),
        bandwidth-not-available    (5)
      },
      other            [6] IMPLICIT Unsigned8  {
        other  (0)
      }
    }
}
UnconfirmedService ::= CHOICE {  -- Kodierung in „Service-Type"
  unsolicited-status [24] IMPLICIT Status-Response,
  re-negotiate       [25] IMPLICIT Re-negotiate,
  event-notification [26] IMPLICIT Event-notification
}
Reject ::= SEQUENCE {  -- Kodierung in „Service-Type"
  original-invoke-id [0]  IMPLICIT InvokeID,
  reject-code        [1]  IMPLICIT Unsigned8  {
    other              (0),
    deadline-violation (1),
    pdu-size           (2)
  }
}
InitiatePDU ::= CHOICE { -- Kodierung in „Service-Type"
  initiate-request   [0]  IMPLICIT Initiate-RequestPDU,
  initiate-response  [1]  IMPLICIT Initiate-ResponsePDU,
  initiate-error     [2]  IMPLICIT Initiate-ErrorPDU
}
Initiate-RequestPDU ::= SEQUENCE {
  rtsims-features-supported-calling[0]  IMPLICIT BIT STRING
  -- Belegung der Bitposition entsprechend des Tags der Dienste
  -- Position 0-26: als Requester, Position 27-52: als Responder
}
Initiate-ResponsePDU ::= SEQUENCE {
  rtsims-features-supported-called [1]  IMPLICIT BIT STRING
  -- Belegung der Bitposition entsprechend des Tags der Dienste
  -- Position 0-26: als Requester, Position 27-52: als Responder
}
Initiate-ErrorPDU ::= SEQUENCE {
  error-code                          [0]  IMPLICIT Unsigned8  {
    user-denied              (0),
    features-not-supported   (1),
    other                    (2)
  },
  rtsims-features-supported-called [1]  IMPLICIT BIT STRING
  -- Belegung der Bitposition entsprechend des Tags der Dienste
  -- Position 0-26: als Requester, Position 27-52: als Responder
}
Create-Request ::= SEQUENCE {
  object-type      [0]  IMPLICIT Object-type,
  object-id        [1]  IMPLICIT Object-ID,
  list-of-domains  [2]  SEQUENCE OF SEQUENCE {
    domain-id    [0]  Object-ID
  } OPTIONAL
}
Create-Response ::= NULL
Delete-Request ::= SEQUENCE {
  object-type    [0]  IMPLICIT Object-type,
  object-id      [1]  IMPLICIT Object-ID
}
Delete-Response ::= NULL
Status-Request ::= SEQUENCE {
  object-type [0]  IMPLICIT Object-type,
```

```
  object-id   [1]  IMPLICIT Object-ID
}
Status-Response ::= SEQUENCE {
  time            [0]  IMPLICIT Timestamp,
  sync-quality    [1]  SyncQuality,
  status-info     [2]  CHOICE {
    rtvfd-status              [0]  IMPLICIT SEQUENCE {
      logical-status  [0]  IMPLICIT Unsigned8  {
        state-changes-allowed       (0),
        no-state-changes-allowed    (1),
        limited-services-permitted  (2),
        support-services-allowed    (3)
      },
      physical-status [1] IMPLICIT Unsigned8  {
        operational          (0),
        partially-operational(1),
        inoperable           (2),
        needs-commissioning  (3)
      }
    },
    program-invocation-status [1]  IMPLICIT Unsigned8  {
        loading  (0),
        idle     (1),
        running  (2)
    },
    scheduler-status          [2]  IMPLICIT Unsigned8  {
        empty    (0),
        loading  (1),
        stopped  (2),
        stopping (3),
        preparing(4),
        running  (5)
    },
    vmmd-status               [3]  SEQUENCE {
      op-status               [0]  IMPLICIT Unsigned8  {
        operational   (0),
        busy          (1),
        not-accessible(2)
      },
      current-settings          [1]  SEQUENCE {
        coding-technique [0]  Coding-technique, -- nach Geraetedatenblatt
        coding-extension [1]  OCTET STRING,     -- nach Geraetedatenblatt
        x-position       [2]  FloatingPoint,
        y-position       [3]  FloatingPoint,
        z-position       [4]  FloatingPoint,
        b-position       [5]  FloatingPoint,
        c-position       [6]  FloatingPoint
      },
      current-extended-settings [2]  SEQUENCE {
        zoom          [0]  INTEGER,
        focus-mode    [1]  INTEGER,
        shutter-speed [2]  FloatingPoint
      } OPTIONAL
    }
  },

  local-detail  [3]  IMPLICIT OCTET STRING -- Laengenbegrenzung: 5 byte
}
Get-attributes-Request ::= SEQUENCE {
  object-type               [0]  IMPLICIT Object-type,
  object-id                 [1]  IMPLICIT Object-ID
}
Get-attributes-Response ::= SEQUENCE {
  rtsims-deletable   [0]  IMPLICIT BOOLEAN,
  description        [1]  CHOICE {
    rtvfd-description  [0]  Object-ID,
```

```
    domain-description [1]  INTEGER,
    pi-description     [2]  SEQUENCE OF Object-ID,
    vmmd-description   [3]  SEQUENCE {
      list-of-coding-techniques [0]  SEQUENCE OF SEQUENCE{
        coding-technique   [0]  Coding-technique, -- nach Geraetedatenblatt
        coding-extension   [1]  OCTET STRING      -- nach Geraetedatenblatt
      },
      movement-range   [1]  SEQUENCE {
        x-direction [0]  Position-interval,
        y-direction [1]  Position-interval,
        z-direction [2]  Position-interval,
        b-angle     [3]  Angle-interval,
        c-angle     [4]  Angle-interval
      },
      cd-description   [2]  SEQUENCE {
        zoom-range  [0]  Position-interval,
        focus-modes [1]  IMPLICIT Unsigned8  {
          auto   (0),
          manual (1),
          both   (2)
        } OPTIONAL,
        available-shutter-speeds  [3]  SEQUENCE OF FloatingPoint
      }
    }
  }
}

Set-Request ::= SEQUENCE {
  object-id      [0]  IMPLICIT Object-ID,
  focus-mode     [1]  IMPLICIT INTEGER OPTIONAL,
  shutter-speed [2]  IMPLICIT FloatingPoint OPTIONAL,
  position       [3]  SEQUENCE {
    x-position     [0]  FloatingPoint,
    y-position     [1]  FloatingPoint,
    z-position     [2]  FloatingPoint,
    b-position     [3]  FloatingPoint,
    c-position     [4]  FloatingPoint
  } OPTIONAL,
  speed-profile [4]  INTEGER
}

Set-Response ::= NULL

Access-control-Request ::= SEQUENCE {
  object-id [0]     IMPLICIT Object-ID,
  mode      [1]     IMPLICIT Unsigned8  {
    request-control  (0),
    release-control  (1)
  }
}

Access-control-Response ::= NULL

Take-image-Request ::= SEQUENCE {
  object-id        [0]  IMPLICIT Object-ID,
  coding-technique [1]  IMPLICIT Coding-technique,
  coding-extension [2]  IMPLICIT OCTET STRING,
  auto-upload      [3]  IMPLICIT BOOLEAN,
  delete-after     [4]  IMPLICIT BOOLEAN
}

Take-image-Response ::= SEQUENCE {
  domain-id [0]  IMPLICIT Object-ID
}

Subscribe-cmd-Request ::= SEQUENCE {
  object-id      [0]  IMPLICIT Object-ID,
  coding-technique [1]  IMPLICIT Coding-technique,
  desired-qualitiy [2]  IMPLICIT OCTET STRING,
  tolerable-range  [3]  SEQUENCE OF OCTET STRING
}
```

```
Subscribe-cmd-Response ::= SEQUENCE {
  stream-id   [0]  IMPLICIT Unsigned16
}

Unsubscribe-cmd-Request ::= SEQUENCE {
  object-id        [0]  IMPLICIT Object-ID,
  stream-id        [1]  IMPLICIT OCTET STRING,
  tolerable-range  [2]  SEQUENCE OF OCTET STRING
}

Unsubscribe-cmd-Response ::= NULL

Coder-control-Request ::= SEQUENCE {
  object-id   [0]  IMPLICIT Object-ID,
  command     [1]  IMPLICIT OCTET STRING
}

Coder-control-Response ::= NULL

Identify-Request ::= NULL

Identify-Response ::= SEQUENCE {
  vendor-name [0]  IMPLICIT VisibleString,
  model-name  [1]  IMPLICIT VisibleString,
  revision    [2]  IMPLICIT VisibleString
}

Load-table-Request ::= Object-ID  -- Bezeichnung der Schedulingtabelle

Load-table-Response ::= NULL

Start-Request ::= NULL

Start-Response ::= NULL

Stop-Request ::= NULL

Stop-Response ::= NULL

Read-Request ::= Object-ID  -- Bezeichnung der Variablen

Read-Response ::= NULL

RT-Read-Request ::= Object-ID -- Bezeichnung der Echtzeitvariablen

RT-Read-Response ::= NULL

Initiatedownloadseq-Request ::= Object-ID

Initiatedownloadseq-Response ::= NULL

Downloadsegment-Request ::= Object-ID

Downloadsegment-Response ::= SEQUENCE {
  data          [0]  IMPLICIT OCTET STRING,
  more-follows  [1]  IMPLICIT BOOLEAN
}

Terminatedownloadseq-Request ::= SEQUENCE {
  domain-id     [0]  IMPLICIT Object-ID,
  final-result  [1]  IMPLICIT BOOLEAN
}

Terminatedownloadseq-Response ::= NULL

Initiateuploadseq-Request ::= Object-ID

Initiateuploadseq-Response ::= NULL

Uploadsegment-Request ::= Object-ID

Uploadsegment-Response ::= SEQUENCE {
  data          [0]  IMPLICIT OCTET STRING,
  more-follows  [1]  IMPLICIT BOOLEAN
}

Terminateuploadseq-Request ::= Object-ID

Terminateuploadseq-Response ::= NULL

Requestdomaindownload-Request ::= Object-ID

Requestdomaindownload-Response ::= NULL
```

```
Requestdomainupload-Request ::= Object-ID

Requestdomainupload-Response ::= NULL

Re-negotiate ::= SEQUENCE {
   stream-id      [0]  IMPLICIT Unsigned16,
   new-qualitiy   [1]  IMPLICIT OCTET STRING
}
Event-notification ::= SEQUENCE {
   variable-id    [0]  IMPLICIT Object-ID,
   event-id       [1]  IMPLICIT Unsigned32,
   time           [2]  IMPLICIT Timestamp,
   sync-qualtity  [3]  IMPLICIT SyncQuality
}
Position-interval ::= SEQUENCE {
   upper-bound [0]  IMPLICIT FloatingPoint,
   lower-bound [1]  IMPLICIT FloatingPoint
}
Angle-interval ::= SEQUENCE {
       upper-bound [0]  IMPLICIT FloatingPoint,
       lower-bound [1]  IMPLICIT FloatingPoint
}
SyncQuality ::= BOOLEAN  -- In Osiris nur zweiwertig
Integer8 ::= INTEGER (-128..127)
Integer16 ::= INTEGER (-32768..32767)
Integer32 ::= INTEGER (-2147483648..2147483647)
Unsigned8 ::= INTEGER (0..255)
Unsigned16 ::= INTEGER (0..65535)
Unsigned32 ::= INTEGER (0..4294967295)
TimeStamp ::= Unsigned32
FloatingPoint ::= OCTET STRING -- 4 Oktette gemaess PROFIBUS
Object-ID ::= Unsigned32
Object-type ::= IMPLICIT Unsigned8 {
   rtvfd              (0),
   domain             (1),
   scheduler          (2)
   scheduling-table   (3),
   program-invocation (4),
   variable           (5),
   rt-variable        (6),
   event-condition    (7),
   vmmd               (8)
}

Coding-technique ::= IMPLICIT Integer16 { -- ggf. zu erweitern
   jpeg                 (0), -- Der Wertebereich von 0 bis 999 ist für
   tiff                 (1), -- Verfahren zur Kodierung von Standbildern
   gif                  (2), -- reserviert.
   targa                (3),
   yuv                  (4),
   other-still-image    (999),
   h-261                (1000),  -- Der Wertebereich von 1000 bis 1999 ist
   mpeg                 (1001),  -- für Verfahren zur Kodierung von Bewegt-
   mjpeg                (1002),  -- bilder reserviert.
   dvi                  (1003),
   avi                  (1004),
   quicktime            (1005),
   other-motion-picture (1999),
   mpeg-audio           (2000),  -- Der Wertebereich von 2000 bis 2999 ist
   pcm                  (2001),  -- für Verfahren zur Kodierung von Audio-
   dpcm                 (2002),  -- informationen reserviert.
   adpcm                (2003),
   other-audio          (2999),
   other                (9999)
}

END  -- Ende der RTSIMS-PDU Spezifikation
```

Anhang D Ergänzungen zum Systemmanagement

Managementinformationen werden in unterschiedlichen Übertragungsrahmen ausgetauscht: Zum einen in einem besonderen Rahmen von DQDFB, dem periodischen Managementrahmen, und zum anderen als Nachrichten zwischen Manager und Agenten. Der Aufbau der periodisch ausgetauschten Protokolldateneinheit ist in Bild D.1 dargestellt.

BIF (1 Bit)	NCF (3 Bit)	reserviert (4 Bit)	LSA (1 Byte)	RQH (2 Byte)	RQL (2 Byte)	TIME (4 Byte)	RTIME (4 Byte)

Byte 0 - 14

IA (1 Byte)	EA (1 Byte)	PSA (1 Byte)	BKS-ID (1 Byte)	CRC8 (1 Byte)	reserviert (16 Byte)

Byte 15 - 34

(a) Periodisch ausgetauschter Management-Rahmen

BIF	Bedeutung
0	Bus A
1	Bus B

(b) Busidentifikation (BIF, Bus-Identifier)

CF (1 Bit)	IS (1 Bit)	ES (1 Bit)

(c) Anzeige der Netzwerkkonfiguration (NCF, Network Configuration Field)

Bild D.1: Format der periodisch ausgetauschten Protokolldateneinheiten des Systemmanagements

Die Busidentifikation zeigt an, über welchen Bus die Dateneinheit ausgetauscht wird. Ein gesetztes CF-Bit im NCF-Feld zeigt an, daß das System im Produktivdatenbetrieb genutzt werden kann. Ist es nicht gesetzt, so befindet sich das System noch in der Konfigurationsphase. Mittels des „Include Station" (IS)-Bits kann eine Station anzeigen, daß sie in das System als aktive Station aufgenommen werden möchte. Das „Exclude Station" (ES)-Bit kann von einer Station dazu genutzt werden, das Ausscheiden ihrer selbst oder einer anderen Station anzuzeigen. Das „Last-Station-Address"-Feld enthält die Adresse der vorhergehenden Station und wird vor der Weiterleitung des Rahmens mit der Adresse der lokalen Station besetzt. Gleiches gilt für die Kopien der Request-Zähler für hoch- bzw. niederpriore Anforderungen („RQH" bzw. „RQL") auf diesem Bus. Diese werden von einer Station jedoch nur dann gesetzt, wenn sie Bestandteil der aktuellen Konfiguration ist. In dem Feld „TIME" und „RTIME" überträgt der Manager den aktuellen Stand der Uhr. Die doppelte Übertragung geschieht aus Gründen der Redundanz. Bei gesetztem IS-Bit ist die Adresse der aufzunehmenden Station im Feld „IA" (Include Address) enthalten. Die Adresse einer aus der Konfiguration auszugliederndern Station ist entsprechend im Feld „EA" (Exclude Address) enthalten. In beiden Fällen ist die Adresse der Vorgängerstation im Feld „PSA" (Preceeding Station Address) enthalten. Die Protokolldateneinheit wird durch die bereits in Kapitel 5 eingeführte Blockprüfsumme $P_{CRC8}(x)$ gegen Übertragungsfehler geschützt.

Der Aufbau des während des Anlaufes des Systems gesendeten „Power-Up"-Übertragungsrahmens ist in Bild D.2 dargestellt. Die Adresse der Station wird wiederum redundant eingetragen („SA", Station Address bzw. „RSA" Redundant SA).

SA (1 Byte)	RSA (1 Byte)	reserved (30 Byte)

Bild D.2: Power-Up Payload

Für die als Nachrichten ausgetauschten Protokolldateneinheiten gelten die Festlegungen bezüglich der Struktur der Protokolldateneinheiten, wie sie bereits für RTSIMS getroffen wurden. Die

„Reject"- und „Initiate"-PDU-Typen werden nicht benötigt. In der abstrakten Syntaxnotation ASN.1 sind die OMP-Protokolldateneinheiten wie folgt spezifiziert:

```
OMPPDU DEFINITIONS ::= BEGIN

OMP-PDU ::= CHOICE {
   confirmed-OMPRequestPDU    [1]  IMPLICIT  Confirmed-OMPRequestPDU,
   confirmed-OMPResponsePDU   [2]  IMPLICIT  Confirmed-OMPResponsePDU,
   confirmed-OMPErrorPDU      [3]  IMPLICIT  Confirmed-OMPErrorPDU,
   unconfirmedOMPPDU          [4]  IMPLICIT  Unconfirmed-OMPPDU
}

Confirmed-OMPRequestPDU ::= SEQUENCE {
   invoke-id                  [0]  InvokeID,
   confirmedOMPServiceRequest  [1]  ConfirmedOMPServiceRequest
}

Confirmed-OMPResponsePDU ::= SEQUENCE {
   invoke-id                    [0]  InvokeID,
   confirmedOMPServiceResponse [1]  ConfirmedOMPServiceResponse
}

Confirmed-OMPErrorPDU ::= SEQUENCE {
   invoke-id        [0]       InvokeID,
   ompserviceError    [1]       OMPServiceError
}

Unconfirmed-OMPPDU ::= SEQUENCE {
   invoke-id              [0]  InvokeID,
   unconfirmedOMPService[1]  UnconfirmedOMPService
}

ConfirmedOMPServiceRequest ::= CHOICE {
   sm-reset            [0]  IMPLICIT  SM-Reset-Request,
   sm-cread            [1]  IMPLICIT  SM-Cread-Request,
   sm-cwrite           [2]  IMPLICIT  SM-Cwrite-Request,
   sm-bandwidth        [3]  IMPLICIT  SM-Bandwidth-Request,
   sm-download-table   [4]  IMPLICIT  SM-DownloadTable-Request
}

ConfirmedOMPServiceResponse ::= CHOICE {
   sm-reset            [0]  IMPLICIT  SM-Reset-Response,
   sm-cread            [1]  IMPLICIT  SM-Cread-Response,
   sm-cwrite           [2]  IMPLICIT  SM-Cwrite-Response,
   sm-bandwidth        [3]  IMPLICIT  SM-Bandwidth-Response,
   sm-download-table   [4]  IMPLICIT  SM-DownloadTable-Response
}

UnconfirmedOMPService ::= CHOICE {
   sm-init-clock           [0]  IMPLICIT  SM-InitClock,
   sm-start-clock          [1]  IMPLICIT  SM-StartClock,
   sm-set-clock            [2]  IMPLICIT  SM-SetClock,
   sm-fault-notification   [2]  IMPLICIT  SM-FaultNotification
}

SM-Reset-Request ::= SEQUENCE {
   name              [0]  Name,
   station-address   [1]  Station-address
}

SM-Reset-Response ::= NULL

SM-Cread-Request ::= SEQUENCE {
   counterID         [0]  Name,
   station-address   [1]  Station-address
}

SM-Cread-Response ::= SEQUENCE {
   counterID      [0]  Name,
   counterValue   [1]  Integer32
}

SM-Cwrite-Request ::= SEQUENCE {
   counterID         [0]  Name,
```

```
  station-address  [1]  Station-address
  counterValue     [2]  Integer32
}
SM-Cwrite-Response ::= NULL
SM-FaultNotification ::= SEQUENCE {
  CHOICE {
    physical-fault   [0]  IMPLICIT INTEGER {
      bus-A-down  (0),
      bus-B-down  (1),
      other       (2)
    },
    datalink-fault   [1]  IMPLICIT INTEGER {
      rq-A-low-overflow  (0),
      rq-A-high-overflow (1),
      rq-B-low-overflow  (2),
      rq-B-high-overflow (3),
      qa-low-A-overflow  (4),
      qa-high-A-overflow (5),
      qa-low-B-overflow  (6),
      qa-high-A-overflow (7),
      invalid-var-id     (8)
    },
      management-fault [2]  IMPLICIT INTEGER {
           address-duplication  (0),
           new-predecessor      (1),
           local-time-adjust    (2)
           HOB-absent           (3)
      }
  },
  occurrenceTime          [3]  IMPLICIT TimeStamp,
  synchronisationQuality  [4]  IMPLICIT SyncQuality,
  additionalInformation   [5]  IMPLICIT VisibleString
}
SM-DownloadTable-Request ::= SEQUENCE {
  station-address  [0]  Station-address
  bus-A-id         [1]  IMPLICIT INTEGER,
  list-of-stations [2]  IMPLICIT List-of-stations,
  bus-B-id         [3]  IMPLICIT INTEGER,
  list-of-stations [4]  IMPLICIT List-of-stations
}
bus-A-id INTEGER ::= 0
bus-B-id INTEGER ::= 1
SM-DownloadTable-Response ::= NULL
SM-Bandwidth-Request ::= SEQUENCE {
  source-address[0]  IMPLICIT Station-address,
  bufferID       [1]  IMPLICIT BufferID,
  rate           [2]  IMPLICIT FloatingPoint,
  mode           [3]  IMPLICIT INTEGER {
  create-disabled  (0),
  create-enabled   (1),
  delete-disable   (2),
  delete-totally   (3)
  }
}
SM-Bandwidth-Response ::= Station-address
OMPServiceError ::= CHOICE {
  sm-reset           [0]  IMPLICIT Error-Type,
  sm-cread           [1]  IMPLICIT Error-Type,
  sm-cwrite          [2]  IMPLICIT Error-Type,
  sm-bandwidth       [3]  IMPLICIT Error-Type,
  sm-download-table  [4]  IMPLICIT Error-Type
}
Error-Type ::= SEQUENCE {
  error-class      [0]  IMPLICIT Error-Class,
```

```
  additional-info [1] IMPLICIT VisibleString OPTIONAL
}
Error-Class ::= CHOICE {
  resource [0] IMPLICIT INTEGER {
    other           (0),
    object-undefined (1),
    already-existing (2)
  },
  service  [1] IMPLICIT INTEGER {
    other                    (0),
    object-state-conflict    (1),
    object-constraint-conflict (2),
    illegal-parameter        (3),
    parameter-mismatch       (4)
  },
  access   [2] IMPLICIT INTEGER {
    other                    (0),
    object-access-denied     (1),
    invalid-address          (2),
    object-access-unsupported (3),
    type-conflict            (4)
  },
  other    [3] IMPLICIT INTEGER {
    other (0)
  }
}
SM-InitClock ::= NULL
SM-StartClock ::= NULL
SM-SetClock ::= TimeStamp
List-of-stations ::= SEQUENCE OF Station-address
Station-address ::= Unsigned8
BufferID ::= Unsigned8
SyncQuality ::= BOOLEAN  -- In Osiris nur zweiwertig
Integer8 ::= INTEGER (-128..127)
Integer16 ::= INTEGER (-32768..32767)
Integer32 ::= INTEGER (-2147483648..2147483647)
Unsigned8 ::= INTEGER (0..255)
Unsigned16 ::= INTEGER (0..65535)
Unsigned32 ::= INTEGER (0..4294967295)
TimeStamp ::= Unsigned32
FloatingPoint ::= OCTET STRING -- 4 Oktette gemaess PROFIBUS
InvokeID ::= Unsigned8
Name ::= VisibleString
END  -- Ende der OMPPDU Spezifikation
```

Lebenslauf

Michael Solvie
geb. am 30.11.1964 in Hamburg,
verheiratet, 3 Kinder

1971 – 1975	Grundschule Vizelinstraße in Hamburg
1975 – 1979	Gymnasium Brehmweg in Hamburg
1979 – 1984	Albrecht-Thaer-Gymnasium in Hamburg Abschluß Abitur: Juni 1984
7/84 – 9/84	Grundwehrdienst
9/84 – 6/86	Soldat auf Zeit (Reserveoffiziersanwärter)
10/86 – 6/91	Studium der Informatik an der Friedrich-Alexander-Universität Erlangen-Nürnberg
6/91	Abschluß: Diplom-Informatiker (Univ.) Note: Sehr gut mit Auszeichnung
7/91 – 9/95	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Fertigungsautomatisierung und Produktionssystematik (Prof. Dr.-Ing. K. Feldmann), dabei
4/93 – 9/95	Oberingenieur und Leiter der Gruppe für „Steuerungs- und Sensortechnik"
seit 10/95	Mitarbeiter bei der Siemens AG, Bereich Verkehrstechnik

Reihe Fertigungstechnik Erlangen

Band 1
Andreas Hemberger
Innovationspotentiale in der rechnerintegrierten Produktion durch wissensbasierte Systeme
208 Seiten, 107 Bilder. 1988. Kartoniert.

Band 2
Detlef Classe
Beitrag zur Steigerung der Flexibilität automatisierter Montagesysteme durch Sensorintegration und erweiterte Steuerungskonzepte
194 Seiten, 70 Bilder. 1988. Kartoniert.

Band 3
Friedrich-Wilhelm Nolting
Projektierung von Montagesystemen
201 Seiten, 107 Bilder, 1 Tabelle. 1989.
Kartoniert.

Band 4
Karsten Schlüter
Nutzungsgradsteigerung von Montagesystemen durch den Einsatz der Simulationstechnik
177 Seiten, 97 Bilder. 1989. Kartoniert.

Band 5
Shir-Kuan Lin
Aufbau von Modellen zur Lageregelung von Industrierobotern
168 Seiten, 46 Bilder. 1989. Kartoniert.

Band 6
Rudolf Nuss
Untersuchungen zur Bearbeitungsqualität im Fertigungssystem Laserstrahlschneiden
206 Seiten, 115 Bilder, 6 Tabellen. 1989. Kartoniert.

Band 7
Wolfgang Scholz
Modell zur datenbankgestützten Planung automatisierter Montageanlagen
194 Seiten, 89 Bilder. 1989. Kartoniert.

Band 8
Hans-Jürgen Wißmeier
Beitrag zur Beurteilung des Bruchverhaltens von Hartmetall-Fließpreßmatrizen
179 Seiten, 99 Bilder, 9 Tabellen. 1989. Kartoniert.

Band 9
Rainer Eisele
Konzeption und Wirtschaftlichkeit von Planungssystemen in der Produktion
183 Seiten, 86 Bilder. 1990. Kartoniert.

Band 10
Rolf Pfeiffer
Technologisch orientierte Montageplanung am Beispiel der Schraubtechnik
216 Seiten, 102 Bilder, 16 Tabellen. 1990. Kartoniert.

Band 11
Herbert Fischer
Verteilte Planungssysteme zur Flexibilitätssteigerung der rechnerintegrierten Teilefertigung
201 Seiten, 82 Bilder. 1990. Kartoniert.

Band 12
Gerhard Kleineidam
CAD/CAP: Rechnergestützte Montagefeinplanung
203 Seiten, 107 Bilder. 1990. Kartoniert.

Band 13
Frank Vollertsen
Pulvermetallurgische Verarbeitung eines übereutektoiden verschleißfesten Stahls
XIII + 217 Seiten, 67 Bilder, 34 Tabellen. 1990. Kartoniert.

Band 14
Stephan Biermann
Untersuchungen zur Anlagen- und Prozeßdiagnostik für das Schneiden mit CO_2-Hochleistungslasern
VIII + 170 Seiten, 93 Bilder, 4 Tabellen. 1991. Kartoniert.

Band 15
Uwe Geißler
Material- und Datenfluß in einer flexiblen Blechbearbeitungszelle
124 Seiten, 41 Bilder, 7 Tabellen. 1991. Kartoniert.

Band 16
Frank Oswald Hake
Entwicklung eines rechnergestützten Diagnosesystems für automatisierte Montagezellen
XIV + 166 Seiten, 77 Bilder. 1991. Kartoniert.

Band 17
Herbert Reichel
Optimierung der Werkzeugbereitstellung durch rechnergestützte Arbeitsfolgenbestimmung
198 Seiten, 73 Bilder, 2 Tabellen. 1991. Kartoniert.

Band 18
Josef Scheller
Modellierung und Einsatz von Softwaresystemen für rechnergeführte Montagezellen
198 Seiten, 65 Bilder. 1991. Kartoniert.

Band 19
Arnold vom Ende
Untersuchungen zum Biegeumformen mit elastischer Matrize
166 Seiten, 55 Bilder, 13 Tabellen. 1991. Kartoniert.

Band 20
Joachim Schmid
Beitrag zum automatisierten Bearbeiten von Keramikguß mit Industrierobotern
XIV + 176 Seiten, 111 Bilder, 6 Tabellen. 1991. Kartoniert.

Band 21
Egon Sommer
Multiprozessorsteuerung für kooperierende Industrieroboter in Montagezellen
188 Seiten, 102 Bilder. 1991. Kartoniert.

Band 22
Georg Geyer
Entwicklung problemspezifischer Verfahrensketten in der Montage
192 Seiten, 112 Bilder. 1991. Kartoniert.

Band 23
Rainer Flohr
Beitrag zur optimalen Verbindungstechnik in der Oberflächenmontage (SMT)
186 Seiten, 79 Bilder. 1991. Kartoniert.

Band 24
Alfons Rief
Untersuchungen zur Verfahrensfolge Laserstrahlschneiden und -schweißen in der Rohkarosseriefertigung
VI + 145 Seiten, 58 Bilder, 5 Tabellen. 1991. Kartoniert.

Band 25
Christoph Thim
Rechnerunterstützte Optimierung von Materialflußstrukturen in der Elektronikmontage durch Simulation
188 Seiten, 74 Bilder. 1992. Kartoniert.

Band 26
Roland Müller
CO_2-Laserstrahlschneiden von kurzglasverstärkten Verbundwerkstoffen
141 Seiten, 107 Bilder, 4 Tabellen. 1992. Kartoniert.

Band 27
Günther Schäfer
Integrierte Informationsverarbeitung bei der Montageplanung
195 Seiten, 76 Bilder. 1992. Kartoniert.

Band 28
Martin Hoffmann
Entwicklung einer CAD/CAM-Prozeßkette für die Herstellung von Blechbiegeteilen
149 Seiten, 89 Bilder. 1992. Kartoniert.

Band 29
Peter Hoffmann
Verfahrensfolge Laserstrahlschneiden und -schweißen : Prozeßführung und Systemtechnik in der 3D-Laserstrahlbearbeitung von Blechformteilen
186 Seiten, 92 Bilder, 10 Tabellen. 1992. Kartoniert.

Band 30
Olaf Schrödel
Flexible Werkstattsteuerung mit objektorientierten Softwarestrukturen
180 Seiten, 84 Bilder. 1992. Kartoniert.

Band 31
Hubert Reinisch
Planungs- und Steuerungswerkzeuge zur impliziten Geräteprogrammierung in Roboterzellen
XI + 212 Seiten, 112 Bilder. 1992. Kartoniert.

Band 32
Brigitte Bärnreuther
Ein Beitrag zur Bewertung des Kommunikationsverhaltens von Automatisierungsgeräten in flexiblen Produktionszellen
XI + 179 Seiten, 71 Bilder. 1992. Kartoniert.

Band 33
Joachim Hutfless
Laserstrahlregelung und Optikdiagnostik in der Strahlführung einer CO_2-Hochleistungslaseranlage
175 Seiten, 70 Bilder, 17 Tabellen. 1993. Kartoniert.

Band 34
Uwe Günzel
Entwicklung und Einsatz eines Simulationsverfahrens für operative und strategische Probleme der Produktionsplanung und -steuerung
XIV + 170 Seiten, 66 Bilder, 5 Tabellen. 1993. Kartoniert.

Band 35
Bertram Ehmann
Operatives Fertigungscontrolling durch Optimierung auftragsbezogener Bearbeitungsabläufe in der Elektronikfertigung
XV + 167 Seiten, 114 Bilder. 1993. Kartoniert.

Band 36
Harald Kolléra
Entwicklung eines benutzerorientierten Werkstattprogrammiersystems für das Laserstrahlschneiden
129 Seiten, 66 Bilder, 1 Tabelle. 1993. Kartoniert.

Band 37
Stephanie Abels
Modellierung und Optimierung von Montageanlagen in einem integrierten Simulationssystem
188 Seiten, 88 Bilder. 1993. Kartoniert.

Band 38
Robert Schmidt-Hebbel
Laserstrahlbohren durchflußbestimmender Durchgangslöcher
145 Seiten, 63 Bilder, 11 Tabellen. 1993. Kartoniert.

Band 39
Norbert Lutz
Oberflächenfeinbearbeitung keramischer Werkstoffe mit XeCl-Excimerlaserstrahlung
187 Seiten, 98 Bilder, 29 Tabellen. 1994. Kartoniert.

Band 40
Konrad Grampp
Rechnerunterstützung bei Test und Schulung an Steuerungssoftware von SMD-Bestücklinien
178 Seiten, 88 Bilder. 1995. Kartoniert.

Band 41
Martin Koch
Wissensbasierte Unterstützung der Angebotsbearbeitung in der Investitionsgüterindustrie
169 Seiten, 68 Bilder. 1995. Kartoniert.

Band 42
Armin Gropp
Anlagen- und Prozeßdiagnostik beim Schneiden mit einem gepulsten Nd:YAG-Laser
160 Seiten, 88 Bilder, 7 Tabellen. 1995. Kartoniert.

Band 43
Werner Heckel
Optische 3D-Konturerfassung und on-line Biegewinkelmessung mit dem Lichtschnittverfahren
149 Seiten, 43 Bilder, 11 Tabellen. 1995. Kartoniert.

Band 44
Armin Rothhaupt
Modulares Planungssystem zur Optimierung der Elektronikfertigung
180 Seiten, 101 Bilder. 1995. Kartoniert.

Band 45
Bernd Zöllner
Adaptive Diagnose in der Elektronikproduktion
195 Seiten, 74 Bilder, 3 Tabellen. 1995. Kartoniert.

Band 46
Bodo Vormann
Beitrag zur automatisierten Handhabungsplanung komplexer Blechbiegeteile
126 Seiten, 89 Bilder, 3 Tabellen. 1995. Kartoniert.

Band 47
Peter Schnepf
Zielkostenorientierte Montageplanung
144 Seiten, 75 Bilder. 1995. Kartoniert.

Band 48
Rainer Klotzbücher
Konzept zur rechnerintegrierten Materialversorgung in flexiblen Fertigungssystemen
156 Seiten, 62 Bilder. 1995. Kartoniert.

Band 49
Wolfgang Greska
Wissensbasierte Analyse und Klassifizierung von Blechteilen
144 Seiten, 96 Bilder. 1995. Kartoniert.

Band 50
Jörg Franke
Integrierte Entwicklung neuer Produkt- und Produktionstechnologien für räumliche spritzgegossene Schaltungsträger (3-D MID)
196 Seiten, 86 Bilder, 4 Tabellen. 1995. Kartoniert.

Band 51
Franz-Josef Zeller
Sensorplanung und schnelle Sensorregelung für Industrieroboter
190 Seiten, 102 Bilder, 9 Tabellen. 1995. Kartoniert.

Band 52
Michael Solvie
Zeitbehandlung und Multimedia-Unterstützung in Feldkommunikationssystemen
200 Seiten, 87 Bilder, 35 Tabellen, 1996. Kartoniert.