Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
Dipl.-Ing. Armin Lechler
geboren in Bietigheim-Bissingen

Hauptberichter: Prof. Dr.-Ing. Dr. h.c. Alexander Verl
Mitberichter: Prof. Dr.-Ing. Heinz Wörn
Tag der Einreichung: 27. April 2011
Tag der mündlichen Prüfung: 21. Juli 2011

Institut für Steuerungstechnik der Werkzeugmaschinen
und Fertigungseinrichtungen
der Universität Stuttgart
2011

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen
der Universität Stuttgart und dem Fraunhofer-Institut für
Produktionstechnik und Automatisierung IPA

Herausgeber: Prof. Dr.-Ing. Dr. h.c. A. Verl

Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen

Fraunhofer IPA

Armin Lechler

Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme

Nr. 184

JOST-JETTER VERLAG
Fachverlag · 71296 Heimsheim

D 93

ISBN 978-3-939890-78-2
Jost Jetter Verlag, Heimsheim

Druck: printsystem GmbH, Heimsheim

Geleitwort des Herausgebers

In der Reihe „ISW/IPA Forschung und Praxis“ wird fortlaufend über Forschungsergebnisse des Instituts für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW) und des Fraunhofer Instituts für Produktionstechnik und Automatisierung (IPA) berichtet. Die Institute stehen unter gemeinsamer Leitung von Professor Alexander Verl und beschäftigen sich sowohl im Bereich der Grundlagenforschung als auch im Bereich der angewandten Forschung mit der Weiterentwicklung und Optimierung der Automatisierung in der Produktionstechnik. Fraunhofer IPA und ISW repräsentieren Deutschlands größte Forschungseinrichtung im Bereich der Produktionsautomatisierung.

Besonderes Augenmerk wird auf die Systeme Werkzeugmaschine und Roboter gelegt. Die Arbeiten des ISW konzentrieren sich im Besonderen auf die Bereiche Numerische Steuerungstechnik, Bewegungssteuerung, Planungs- und Leitsysteme, Softwaretechnik, Simulationstechnik und Antriebstechnik. Am Fraunhofer IPA konzentriert man sich im Bereich der Automatisierungstechnik auf die Themen: Servicerobotik, Industrierobotik, Montageautomatisierung, Orthopädiesysteme, Prozessautomatisierung (insbesondere im Lifesciencebereich), Messtechnik, Prüftechnik, Bildverarbeitung sowie die Automatisierung in der Reinst- und Mikroproduktion. Dabei stehen Grundlagenforschung und anwenderorientierte Entwicklung in einem stetigen Austausch, wodurch ein ständiger Technologietransfer zur Praxis sichergestellt wird.

Die Buchreihe erscheint in zwangloser Folge und stützt sich auf Berichte über abgeschlossene Forschungsarbeiten und Dissertationen. Sie soll dem Ingenieur bei der Weiterbildung dienen und ihm Hilfestellungen zur Lösung spezifischer Probleme geben. Für den Studierenden bietet sie eine Möglichkeit zur Wissensvertiefung. Sie bleibt damit unter erweitertem Namen und in der inzwischen dritten Generation in der bewährten Konzeption, die ihr die Vorgänger am ISW, Professor Gottfried Stute (1972 - 1982) und die Professor Günter Pritschow (1984 - 2005), gegeben haben.

Der vorliegende Band beschäftigt sich mit der Konzeption einer funktional einheitlichen Applikationsschnittstelle, die es erlaubt, bussystemunabhängig auf verschiedene Ethernet-basierte Bussysteme zuzugreifen. Auf diese Weise können Applikationen für die Automatisierungstechnik ohne Anpassungen über unterschiedliche Bussysteme kommunizieren. Hierfür werden anhand der Objektorientierten Analyse existierender Ethernet-basierter Bussysteme und deren Applikationsprofile verglichen und daraus eine einheitliche Schnittstelle zwischen der Treiber- und der Applikationsebene abgeleitet.

Der Herausgeber dankt der Druckerei für die drucktechnische Betreuung und dem Jost Jetter Verlag für die Aufnahme der Reihe in sein Lieferprogramm.

A. Verl

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart.

Herrn Prof. Dr.-Ing. Dr. h.c. Alexander Verl danke ich herzlichst für die wohlwollende Unterstützung und die wertvollen Anregungen, die zum Gelingen dieser Arbeit wesentlich beigetragen haben, sowie für die Übernahme des Hauptberichtes.

Herrn Prof. Dr.-Ing. Heinz Wörn danke ich sehr für seine Bereitschaft, den Mitbericht zu übernehmen.

Weiterhin danke ich allen aktuellen und ehemaligen Mitarbeitern des Instituts für die kollegiale und offene Zusammenarbeit und die interessanten Diskussionen. Insbesondere danke ich den Herren Dr.-Ing. Stefan Staudt und Dr.-Ing. Andreas Selig für die kritische Durchsicht meiner Arbeit.

Nicht zuletzt danke ich meiner Frau Jasmin für ihre tatkräftige und moralische Unterstützung während meines Studiums, der Arbeit am Institut und bei der Erstellung dieser Ausarbeitung.

Armin Lechler

Inhaltsverzeichnis

Abkürzungen

AO	Applikationsobjekt
APW	Application Wrapper
AT	Antwort-Telegramm
BDDI	Bus Device Driver Interface
CAM	Computer Aided Manufacturing
CAN	Controller Area Network
CiA	CAN in Automation
CIP	Common Industrial Protocol
CORBA	Common Object Request Broker Architecture
COW	Communication Wrapper
CPF	Communication Profile Families
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DCOM	Distributed Component Object Model
DTM	Device Type Manager
EDDL	Electronic Device Description Language
FDDI	Fieldbus Device Driver Interface
FB	Funktionsblock
FDT	Field Device Tool
FK	Formfräsen/Kontur
FPGA	Field Programmable Gate Array
FSP	Function Specific Profile
GUI	Graphical User Interface
IDN	Identification Number
IFEP	Improved Fast Evolutionary Programming
IRT	Isochronous Real Time
K	Komponente
LDI	Logical Device Interface
M	Modellierungsebene
MAC	Media Access Control
MDT	Master-Daten-Telegramm
MST	Master Synchronisationstelegramm
NC	Numerical Control
NRT	Non Real Time
OCCI	Open Control Communication Interface

OLE	Object Linking and Embedding
OOA	Objektorientierte Analyse
OOD	Objektorientiertes Design
OOP	Objektorientiertes Programmieren
OPC	OLE for Process Control
OSA	Open System Architecture – Platform for Universal Services
OSACA	Open System Architecture for Controls within Automation Systems
OSADL	Open Source Automation Development Lab
OSI	Open Systems Interconnection
PC	Personal Computer
PDS	Power Drive System
PHY	Physikalische Schnittstelle zum Netzwerk
PI	Proportional Integral
PID	Proportional Integral Derivative
PLC	Programmable Logic Control
PTP	Precision Time Protocol
QoS	Quality of Service
RT	Real Time
SCI	SOFIA Component Interface
SERCOS	Serial Realtime Communication System
SI	Système International d'unités
SIL	Safety Integrity Level
SOFIA	Modulares Softwaresystem für intelligente Antriebe
SPS	Speicherprogrammierbare Steuerung
SVC	Service Channel
TCP/IP	Transmission Control Protocol / Internet Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
XML	Extensible Markup Language

Abstract

Highly unstable markets in regard to lot sizes, product life cycles and costs require flexibility and, in particular, variability in automation engineering. The latter has, besides mechanical variations, an impact on general system architecture of machines and units. Current developments of system architecture veer toward decentralization and interlinked mechatronic modules. The trend in automation technology results in a higher communication effort within and between those modules. In order to meet communication requirements, the various bus systems found on the market have been ported to Ethernet and new systems have been developed. Use of standard Ethernet components for industrial communication has not lead to a standardization of the communication systems. Higher bandwidth due to the utilization of Ethernet has only contributed to the fact that there are hardly any unique features anymore in the individual systems.

Specifications of Ethernet-based bus systems are extremely complex and comprehensive, last but not least because of the wide functional range of the application profiles. Training and implementation effort for a software stack that processes communication within an automation component is enormous. In addition, time and cost for configuration and diagnosis are extremely high with the use of Ethernet-based bus systems. Because of high implementation effort and development costs, manufacturers of components have to decide on one or few bus systems which they can support. Besides technical efforts there are also corporate-political interests which get in the way of a mutual consent to one system. For producers and especially for users of bus system components the access to application profiles is of particular interest. Since this access also differs between the bus systems, the bus specific information is contained in the applications. Applications are therefore depending on the used communication system and they cannot communicate via another bus system.

In order to generate automation systems and especially user programs that can be applied independent of the used communication technology, a standardized user interface has to be designed. This interface needs to be functional standard and map all required functions of available communication systems and application profiles. The method presented in this thesis describes the conceptual design of such a functionally standardized user interface.

None of the existing scientific approaches and industrial implementations define a functional bus system-independent interface between application and communication. Deficits of the scientific approaches are essentially the lack of universality in regard to branch and application. Previous comparisons deal mainly with performance of different systems. These comparisons cannot be taken as a basis for a standardization of an interface. In industrial solutions there are none without bus-specific information in the application. For changeover to another bus system global adaptations are necessary. A structured perception of industrial bus systems does also not exist.

This thesis deals in a first step with abstraction of examined bus systems and application profiles first. Here is analyzed which functionalities are provided by individual systems and profiles. Based on the specifications of Ethernet-based bus systems they are identified and in a next step generalized. For this purpose, a joint abstraction level between respective bus systems and application profiles is deduced. Within this joint level of abstraction a structural correlation in form of a classification of the functionalities is derived. This abstraction provides the basis for the following comparison.

In a next step the individual functionalities are examined closer by means of specialization. For this, the functionalities are subdivided into functions. Based on specifications for Ethernet-based bus systems the functions for communication and drive profiles are described in detail. The functions are compared between various bus systems and similarities as well as differences are elaborated.

Based on the previous comparison the functionally a standardized interface is defined in a further step. For this the comparable functions and functionalities are uniformly described and transferred into the standardized interface. Here a method is defined, that decides in particular, which functions and functionalities are described uniformly. In addition to the functional contents of the interface the structure of the interface and its configuration is defined.

Results of this method are validated at a modular woodworking machine. The implementation of these results is analyzed by means of an exemplary application of a woodworking machine at Homag. It can be shown that the deduced functionally standardized application interface can be implemented successfully. With just an exception of nine percent, the functions used for this application can be addressed directly via standardized functions. The application on part of the control is therefore independent from the used

communication system and drive profile. Also producer of used components is irrelevant in the control application.

The concept derived in this thesis for a functionally standardized application interface for Ethernet-based bus systems can be extended by further bus systems and application profiles.

1 Einleitung und Problemstellung

Stark schwankende Märkte hinsichtlich Stückzahlen, Produktlebenszyklen und Kosten von Produkten erfordern Flexibilität und insbesondere Wandlungsfähigkeit in der Automatisierungstechnik /1/. Die Wandlungsfähigkeit der Automatisierungstechnik hat, neben maschinenbaulichen Veränderungen, Auswirkungen auf die generelle Systemarchitektur von Automatisierungssystemen /2, 3/. Entwicklungen der Systemarchitektur gehen in Richtung Dezentralisierung und vernetzter mechatronischer Module /4, 5/.

Der damit verbundene, stetige Anstieg der digitalen Kommunikation zwischen dezentralen Automatisierungskomponenten in der Fertigungstechnik führt zu steigenden Leistungsanforderung an industrielle Kommunikationssysteme. Es existiert eine enge digitale Vernetzung innerhalb und zwischen den einzelnen Steuerungsebenen im industriellen Fertigungsbereich /6, 7, 8/. Die einzelnen Steuerungsebenen (Bild 1.1) unterscheiden sich hinsichtlich der Anforderungen an die Kommunikation.

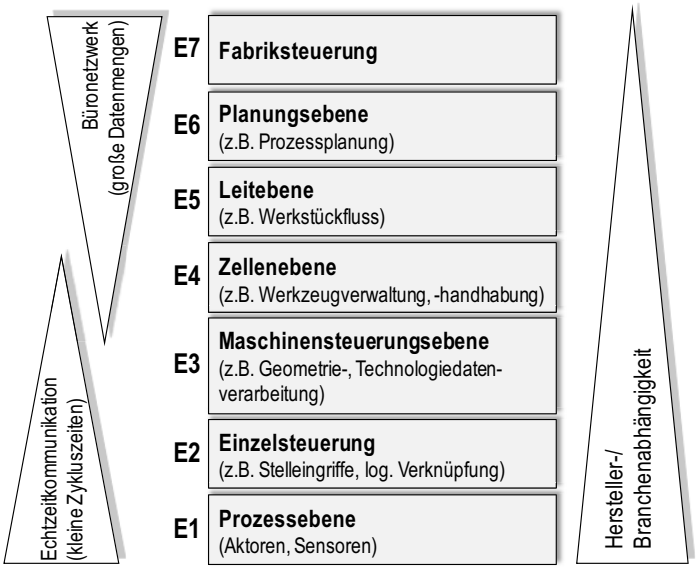


Bild 1.1: Steuerungsebenen und Kommunikation des industriellen Fertigungsbereichs

In den oberen Steuerungsebenen (E7-E4) werden hauptsächlich große Datenmengen (zum Beispiel Zeichnungen, Programme) über das Kommunikationssystem transportiert, die eine geringe Anforderung an das deterministische Übertragungsverhalten stellen. Hier finden meist Büronetzwerke mit hohem Datendurchsatz Anwendung.

In den unteren Steuerungsebenen (E3-E1) sind vor allem kleine Datenmengen (zum Beispiel Prozessabbild, Steuerbefehle) in kurzen Zyklen zu übertragen, deshalb werden hier echtzeitfähige Kommunikationssysteme eingesetzt. Echtzeitfähige industrielle Kommunikationssysteme garantieren ein deterministisches Übertragen von kleinen Datenmengen innerhalb und zwischen den unteren Steuerungsebenen. Zusätzlich dienen sie zur Synchronisation der angebundenen Teilnehmer.

Von der oberen zur unteren Steuerungsebene hin nimmt ebenso die Hersteller- und Branchenabhängigkeit zu /9, 10/. Denn als Mitte der 80er Jahre die Entwicklung der industriellen Kommunikationssysteme, den sogenannten Feldbussen begann, stand die Reduzierung des Verkabelungsaufwands bei der Entwicklung im Vordergrund. Dabei hatten die jeweiligen Branchen spezifische Anforderungen an die industrielle Kommunikation, was Echtzeitfähigkeit und Bandbreite angeht /11/. Der P-Net Bus /12/ beispielsweise, der für die Verfahrenstechnik entwickelt wurde, hatte eine geringere Anforderung an die Zykluszeit beim Übertragen von Temperaturen im Bereich von Sekunden. Sercos /13/ hingegen wurde für die Übertragung von Positionswerten an digitale Antriebe in Werkzeugmaschinen im Interpolationstakt (1-10ms) ausgelegt.

Feldbus	Entwickler	Haupteinsatzbereich					Physik/Buszugriff				
		Verfahrenstechnik	Vernetzung von Aktor/Sensor	Automobil	Synchronisation von digitalen Antrieben	Fertigungs- und Prozessautomatisierung	RS 485 / Token Passing	RS 485 / CSMA/CA	RS 485 / Zeitschlitz	Lichtwellenleiter / Zeitschlitz	anfängliche Datenrate (Mbit/s)
P-NET	PROCESS-DATA	●					●				0,1
	Phoenix Contact		●						●		2
CAN	Bosch / Daimler Benz			●				●			1
	ZVEI und VDW				●					●	4
	Bosch und Siemens					●	●				0,5

CSMA/CA Carrier Sense Multiple Access / Collision Avoidance
VDW Verein Deutscher Werkzeugmaschinenfabriken
ZVEI Zentralverband Elektrotechnik- und Elektroindustrie

Bild 1.2: Historische Entwicklung und Einsatzbereiche der ersten Feldbussysteme

Hinter jedem Feldbussystem standen meist einzelne größere Firmen einer Branche, die ihre Interessen bei der Entwicklung einbrachten. Somit wurde für nahezu jede Branche ein auf die spezifischen Anforderungen angepasstes Feldbussystem entwickelt (Bild 1.2). Diese Feldbussysteme sind jedoch bezüglich der Übertragungsphysik, des Buszugriffes und des Protokollaufbaus inkompatibel zueinander und unterscheiden sich grundlegend.

Der Trend in der Automatisierungstechnik zur Dezentralisierung und zu verteilten, intelligenten mechatronischen Systemmodulen führt zu einem erhöhten Datenaufkommen innerhalb und zwischen solchen Modulen /14/. Dieses Datenaufkommen kann mit den ursprünglichen Feldbussystemen aufgrund ihrer eingeschränkten Bandbreite nicht übertragen werden.

Um diesen Anforderungen gerecht zu werden, wurden ab dem Jahr 2000 die verschiedenen Felsbussysteme, die am Markt existieren, auf das im Bürobereich verbreitete Ethernet /15/ portiert /16, 17, 18/. Die Nutzung von verfügbaren Ethernet-Komponenten sollte neben höherer Bandbreite zusätzlich zu Kostenersparnis durch geringeren Implementierungsaufwand und bereits verfügbarer Hardware führen /19/. Bei der Nutzung der Ethernet-Technologie als industrielle Kommunikation entstanden neben den portierten zusätzliche neue Bussysteme mit wiederum neuen Übertragungsprotokollen. Bild 1.3 stellt eine Auswahl an Bussystemen und deren zeitliche Entwicklung dar.

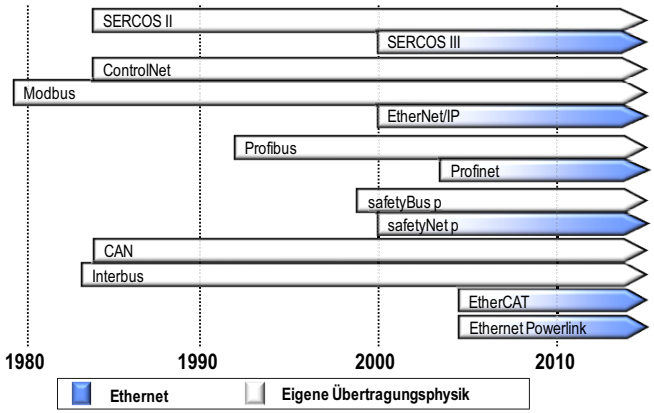


Bild 1.3: Zeitliche Entwicklung ausgewählter Feldbussysteme

Die Ethernet-Technologie ist aufgrund ihres zufälligen Buszuteilungsverfahrens CSMA/CD (Carrier Sense Multiple Access/Collision Detection) nur durch spezielle Erweiterungen für eine deterministische Datenübertragung und Synchronisation geeignet. Deshalb waren bei der Umsetzung existierender und der Definition von neuen Bussystemen Erweiterungen zur Erreichung der Echtzeitfähigkeit notwendig. Diese Erweiterungen sind bei allen Systemen unterschiedlich umgesetzt, so dass weiterhin eine Inkompatibilität zwischen den Ethernet-basierten Bussystemen besteht.

Die höhere Bandbreite durch die Verwendung von Ethernet hat dazu beigetragen, dass es kaum noch technologische Alleinstellungsmerkmale einzelner Systeme gibt. Bussysteme, die beispielsweise bisher hauptsächlich für die Vernetzung von Antrieben zum Einsatz kamen, sogenannte Motionbusse (zum Beispiel Sercos /20/), und sich durch eine kleine Zykluszeit und hohe Synchronisationsgenauigkeit auszeichnen, wurden um Mechanismen (Beispiel E/A-Profil) erweitert, um sinnvoll E/A-Geräte in das Netzwerk einzubinden. Systeme, die eher auf den Bereich der Vernetzung langsamerer E/As ausgelegt waren, sogenannte Feldbusse (Beispiel Profibus /30/), wurden um die benötigte Performance zur Kommunikation mit Antrieben erweitert /21/. Die Anforderungen der Fertigungstechnik nach kurzen Zykluszeiten, hohen Bandbreiten und umfangreichen Applikationsprofilen an das Kommunikationssystem kann mittlerweile durch jedes der existierenden Ethernet-basierten Bussysteme erfüllt werden /22/.

Durch die gestiegene Bandbreite mit dem Einsatz von Ethernet als Übertragungsphysik werden zunehmend mehr echtzeitkritische Informationen zwischen den einzelnen Automatisierungskomponenten zur Verbesserung der Prozessstabilität ausgetauscht. Bei der dezentralen Prozessregelung sind extrem kurze Zykluszeiten bei der Informationsübertragung von Soll- und Istwerten gefordert. Dabei sind teilweise Zykluszeiten bis zum Stromregeltakt (31,25µs) gefordert. /23, 24/. Neben den Soll- und Istwerten zur Prozessregelung /25/ werden zusätzlich weitere Informationen zur Prozess- und Maschinendiagnose ausgetauscht /26, 27/.

Um eine zusätzliche Verkabelung für sicherheitsrelevante Signale bis SIL3 (Safety Integrity Level) /28/ einzusparen, bietet jedes Bussystem auch hierfür eine separate Möglichkeit, diese Informationen parallel zu den nichtsicheren Daten auf demselben Bus zu übertragen /29, 30/.

Die hohe Leistungsfähigkeit und der gestiegene Funktionsumfang der aktuellen Ethernet-basierten Bussysteme haben jedoch nicht dazu geführt, dass sich ein System am Markt durchsetzt. Aufgrund von firmenpolitischen Entscheidungen sind in den letzten Jahren weitere Systeme hinzugekommen und es werden weitere folgen /31/. Bild 1.4 zeigt die Entwicklungen der Bussysteme für die industrielle Kommunikation der letzten Jahre. Dabei wird deutlich, dass bereits 19% der industriellen Kommunikation über Ethernet-basierte Bussysteme erfolgt. Dieser Trend wird sich in den nächsten Jahren weiter fortsetzen.

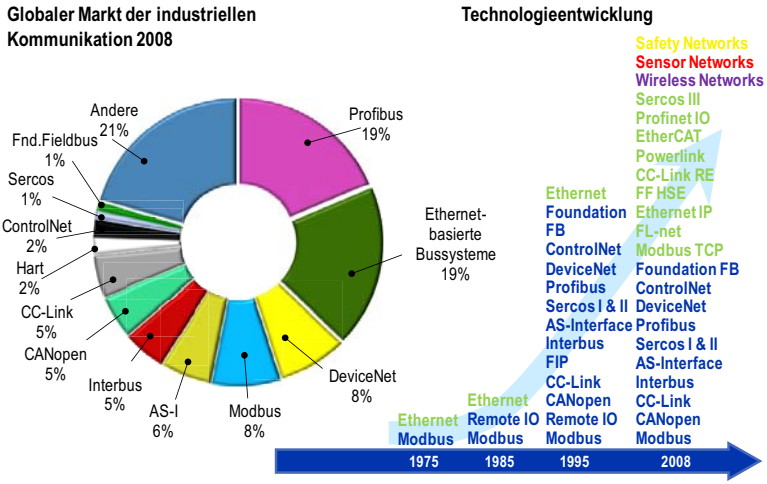
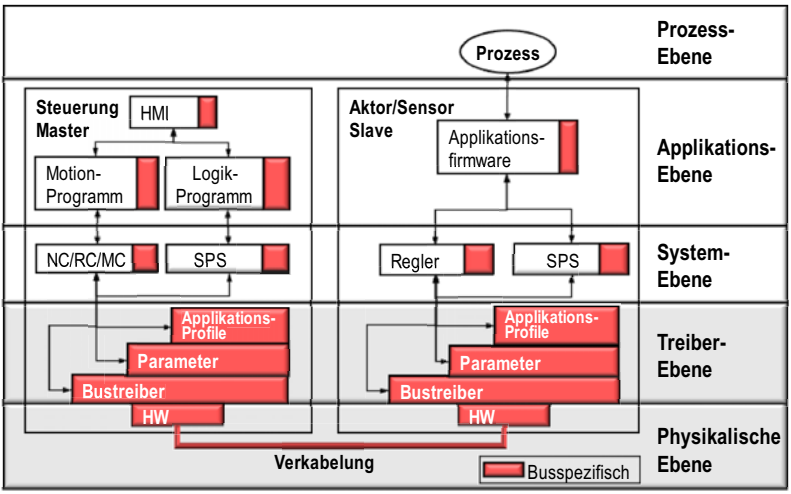


Bild 1.4 Marktanteile der Bussysteme 2008 (Quelle: IMS Research/32/)

Die Spezifikationen von Ethernet-basierten Bussystemen sind extrem komplex und umfangreich, nicht zuletzt durch den großen Funktionsumfang der Applikationsprofile. Der Einarbeitungs- und Implementierungsaufwand für einen Softwarestack, der die Kommunikation innerhalb einer Automatisierungskomponente abwickelt, ist enorm. Zusätzlich ist der Aufwand für die Konfiguration und Diagnose bei der Anwendung von Ethernet-basierten Bussystemen äußerst umfangreich.

Komponentenhersteller müssen sich aufgrund des hohen Implementierungsaufwands und den damit verbundenen Entwicklungskosten für eines oder wenige Bussysteme entscheiden, die sie in ihren Produkten unterstützen können.

Für Hersteller und insbesondere Anwender von Bussystemkomponenten ist der Zugriff auf die Applikationsprofile von besonderem Interesse. Da sich dieser Zugriff ebenfalls zwischen den Bussystemen unterscheidet, führt es dazu, dass busspezifische Informationen in den Applikationen selbst enthalten sind (Bild 1.5). Applikationen sind dadurch vom verwendeten Kommunikationssystem abhängig und können ohne Anpassung nicht über ein anderes Bussystem kommunizieren. Die busspezifischen Umsetzungen finden nicht nur in der physikalischen Ebene und der Treiberebene sondern auch in der System- und Applikationsebene statt. Neben den technischen Aufwänden stehen auch firmenpolitische Interessen hinter den einzelnen Systemen, die eine Einigung auf ein System verhindern.



NC Numerical Control, RC Robot Control, MC Motion Control, HMI Human Machine Interface, SPS Speicherprogrammierbare Steuerung, HW Hardware

Bild 1.5: Busspezifische Abhängigkeiten in Automatisierungskomponenten

Die unterschiedlichen Zugriffe auf die Bussysteme führen dazu, dass im Anwendungsprogramm (Applikation) der Applikationseben des Automatisierungssystems für dessen

korrekte Funktion busspezifische Anweisungen enthalten sein müssen. Bild 1.6 zeigt beispielhaft ein NC-Programm und eine Konfigurationsdatei einer NC-Steuerung, in denen Funktionen bzw. Parameter enthalten sind, die abhängig vom verwendeten Kommunikationssystem sind.

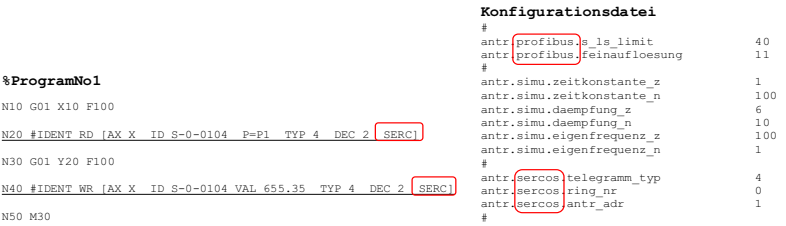
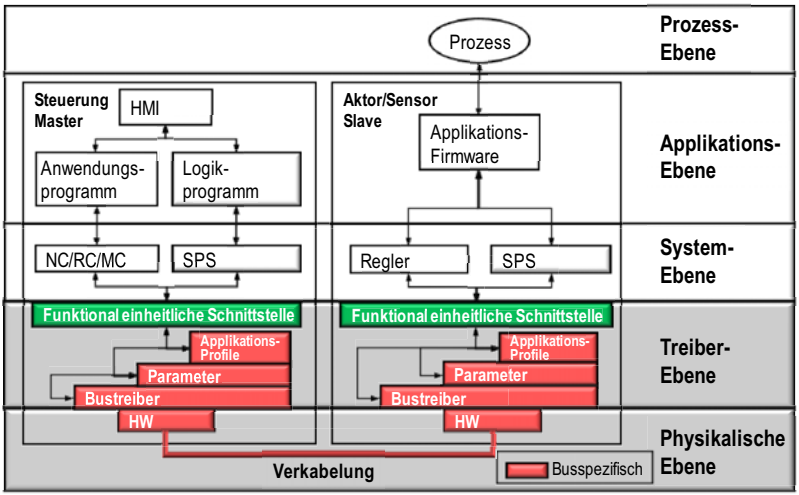


Bild 1.6: Applikationen mit busspezifischem Inhalt
(Quelle: Industrielle Steuerungstechnik GmbH)

Applikationsingenieure haben ihr Wissen in der Anwendungsprogrammierung und der Prozesssteuerung des Automatisierungssystems und nicht in der Kommunikationstechnik. Selbst ein einfaches busspezifisches Anwendungsprogramm ist zum einen nicht auf einem identischen Automatisierungssystem mit einem lediglich anderen Bussystem lauffähig. Zum anderen benötigt ein Anwender bei der Inbetriebnahme oder Instandhaltung Expertenwissen über das jeweilige Bussystem. Dies führt zu hohen Kosten bei der Inbetriebnahme und Wartung.



NC Numerical Control, RC Robot Control, MC Motion Control, HMI Human Machine Interface, SPS Speicherprogrammierbare Steuerung, HW Hardware

Bild 1.7: Funktional einheitliche Applikationsschnittstelle

Um der Problemstellung von busspezifischen Teilen im Anwendungsprogramm zu begegnen, wird in dieser Arbeit eine funktional einheitliche Applikationsschnittstelle definiert, die den Zugriff von der Applikations- und Systemebene auf verschiedene Bussysteme und Applikationsprofile vereinheitlicht (Bild 1.7) /33, 34/. Beim Wechsel des Bussystems bleiben dadurch die Ebenen oberhalb der vereinheitlichten Schnittstelle unberührt. Es werden sowohl die Seite der Steuerung (typischerweise der Kommunikationsmaster), als auch die Seite der Aktoren, bzw. Sensoren (typischerweise die Kommunikationsslaves) betrachtet.

2 Begriffsdefinition und Anforderungen einer funktional einheitlichen Applikationsschnittstelle

In diesem Kapitel werden neben Begriffen Anforderungen an eine funktional einheitliche Applikationsschnittstelle definiert. Diese Anforderungen sollen von der zu definierenden Schnittstelle für den Zugriff von der Applikation auf das Bussystem, sowie die jeweiligen Applikationsprofile erfüllt werden.

2.1 Begriffsdefinition

Im Folgenden werden Begriffe, die relevant für diese Arbeit sind, definiert. Es existieren keine einheitlichen Definitionen der Begriffe und es werden ihnen im Bereich der Kommunikations- und Automatisierungstechnik teilweise unterschiedliche Bedeutungen zugeordnet. Die Zusammenhänge zwischen den Begriffen sind in Bild 2.1 dargestellt. Sowohl Applikationsprofile als auch Ethernet-basierte Bussysteme beschreiben Funktionalitäten. Diese Funktionalitäten setzen sich wiederum aus mehreren Funktionen zusammen, die von der Applikation angesprochen werden können. Die dazu notwendigen Informationen werden über die Kommunikation in Form von Parametern übertragen.

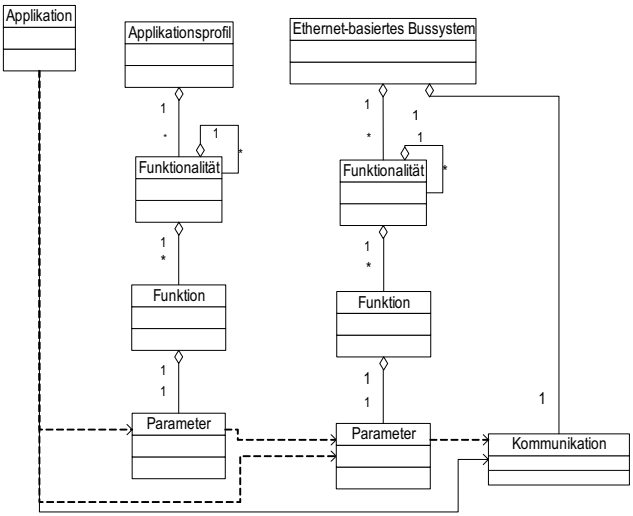


Bild 2.1: Strukturelle Zusammenhang der Begriffsdefinitionen

Die Begriffe definieren sich für diese Arbeit im Einzelnen zu:

Ethernet-basierte Bussysteme/Kommunikation
Es sind hierunter industrielle Echtzeitkommunikationssysteme zu verstehen, die auf dem Ethernet-Standard IEEE 802.3 /15/ basieren und diesen eventuell hinsichtlich des Buszugriffs verändern. Sie übertragen die Daten zwischen den Kommunikationsteilnehmern und deren jeweiligen Applikationen. Hierunter fallen sowohl Feldbusse, als auch Motionbusse. Zwischen diesen wird im Folgenden nicht mehr unterschieden, da wie bereits in Kapitel 1 beschrieben, keine technologische Unterscheidung mehr sinnvoll ist.

Funktionalität
Funktionalitäten sind Teile des Spezifikationsumfangs von Bussystemen und Applikationsprofilen. Sie dienen zur Einteilung der umfangreichen Beschreibung der Bussysteme in logische funktionale Einheiten, um eine spezifikationsübergreifende Strukturierung vornehmen zu können. Beispiel für eine Funktionalität ist das antriebsgeführte Referenzieren eines Servoantriebs.

Funktion/Parameter
Eine Funktion ist ein Teil einer Funktionalität. Einer Funktion ist genau ein Parameter zugeordnet, der über das Bussystem zwischen den Applikationen übertragen werden kann. Beispiel für eine Funktion ist der Parameter der Referenziergeschwindigkeit des antriebsgeführten Referenziervorgangs eines Servoantriebs durch einen Zugriff über das Bussystem.

Applikationsprofil
Als Applikationsprofile werden Spezifikationen bezeichnet, die Funktionalitäten und Funktionen, sowie deren Bedeutung aus Applikationssicht beschreiben. Es existieren einzelne Applikationsprofile für bestimmte Geräteklassen (zum Beispiel elektrische Antriebe, E/As).

Applikation
Eine Applikation beschreibt allgemein das konkrete Verhalten einer Maschine oder Anlage bezüglich ihrer logischen Zusammenhänge und Bewegungsabläufe. Hierzu gehören die Steuerungsprogramme (NC-, SPS-, IRL-Programm, Antriebs-, Klemmenfirmware) der Applikationsebene, die auf den Steuerungsmitteln (CNC, SPS, RC, Antriebsregler, Buskoppler) der Steuerungsebene ausgeführt werden.

Einheitlich
Einheitlich bedeutet in dieser Arbeit, die Zusammenfassung (Vereinheitlichung) der betrachteten Ethernet-basierten Bussysteme und Applikationsprofile bezüglich Ihrer Funktionalitäten und Funktionen sowie deren Zugriff. Vereinheitlichte Funktionalitäten lassen sich auf die busspezifischen Funktionalitäten des jeweiligen Bussystems abbilden.

2.2 Anforderungen

2.2.1 Universalität und breite Umsetzbarkeit

Um mit der Schnittstelle eine möglichst breite funktionale Abdeckung und Umsetzbarkeit zu erreichen, besteht die Anforderung der Universalität. Die Universalität zeichnet sich durch Unabhängigkeit von Hard- und Softwarekomponenten aus. Durch sie soll die bisherige busspezifische Sicht auf industrielle Kommunikationssysteme abstrahiert werden. Die Anforderung der Universalität einer solchen Schnittstelle bezieht sich auf folgende Punkte:

- **Applikationsunabhängigkeit**
  Die Schnittstelle soll nicht nur für eine bestimmte oder eine eingeschränkte Klasse von Applikationen definiert werden. Sie soll möglichst für alle Applikationen und Branchen anwendbar sein, in denen Ethernet-basierte Bussysteme zum Einsatz kommen.
- **Geräteunabhängigkeit**
  Die Ergebnisse sollen nicht auf eine einzige Geräteklasse hin zugeschnitten sein, sondern möglichst viele Geräteklassen aus dem Automatisierungsbereich abdecken. Die Geräteunabhängigkeit soll sich sowohl auf Master- als auch auf Slave-Geräte und deren Applikationen beziehen.
- **Bussystemunabhängigkeit**
  Die Schnittstellengestaltung soll unabhängig vom verwendeten Bussystem bzw. dessen Applikationsprofil sein. Sie muss sich auf jedes der betrachteten Bussysteme abbilden lassen, ohne dessen funktionale Eigenschaften und dessen Leistungsfähigkeit einzuschränken. Das Bussystem und die Applikationsprofile werden nicht verändert. Es soll lediglich eine Abbildung der funktional einheitlichen Schnittstelle auf die jeweiligen Funktionen der Bussysteme und Applikationsprofile stattfinden.

- **Hardwareunabhängigkeit**
  Die verwendete Hardware darf keine Rolle bei der Schnittstellendefinition spielen und muss entsprechend gekapselt werden. Die Schnittstelle muss sich ebenso auf einer PC- wie auf einer Mikrocontrollerarchitektur abbilden lassen können. Dies ist erforderlich, um eine Umsetzung sowohl auf performanten PC-basierten Steuerungen als auch auf rechenschwachen Mikrocontrollern in Slavegeräten zu ermöglichen.
- **Herstellerunabhängigkeit**
  Die in den Spezifikationen der Ethernet-basierten Bussysteme und den zugehörigen Applikationsprofilen beschriebenen Funktionen sind nicht immer ausreichend, um den vollen Funktionsumfang eines Gerätes anzusprechen. Funktionen die über die spezifizierten hinausgehen, sind herstellerspezifisch. Bei der Konzeption einer einheitlichen funktionalen Schnittstelle dürfen keine herstellerspezifischen Funktionen enthalten sein.
- **Vollständigkeit**
  Es müssen alle Funktionalitäten und Funktionen eines Bussystems und Applikationsprofils aus der Applikation heraus erreichbar sein. Die einheitliche Schnittstelle darf keine Einschränkungen der Funktionalitäten hervorrufen.

2.2.2 Einfache Handhabbarkeit

Die Definition einer funktional einheitlichen Applikationsschnittstelle erfordert, dass diese bei der Applikationsentwicklung einfach handhabbar ist. Dazu zählen folgende Anforderungen, die erfüllt werden müssen:

- **Strukturierung**
  Es ist eine übersichtliche Struktur zu definieren. Diese Struktur muss sich an der mechatronischen Betrachtung von Automatisierungskomponenten orientieren. Neben der rein funktionalen Vereinheitlichung der Bussysteme ist die Kapselung der Komplexität solcher Bussysteme durch eine geeignete Struktur als Anforderung zu sehen.
- **Anzahl der Funktionalitäten und Funktionen**
  Die Anzahl der Funktionalitäten und Funktionen der funktional einheitlichen Anwendungsschnittselle soll im Bereich der Funktionsanzahl eines einzelnen Bussystems liegen.

2.2.3 Eindeutigkeit der Funktionalitäten und Funktionen

Die Eindeutigkeit der einzelnen Funktionalitäten und Funktionen sowie deren klare Abgrenzung zueinander sind besonders wichtig. Die vereinheitlichte Funktionalität muss sich unabhängig vom verwendeten Bussystem immer gleich verhalten. Die Eindeutigkeit muss zum einen für den Anwender gegeben sein. So kann er genau nachvollziehen, wie sich die Funktionalität auswirkt, die er aus der Applikation anspricht. Die Schnittstelle darf keinen Einfluss auf das Prozessergebnis haben. Zum anderen muss exakt definiert sein, auf welche busspezifische Funktionalität sich die vereinheitlichte Funktionalität abbildet (Bild 2.2). Im dargestellten Beispiel der Interpolation als Funktionalität müssen sich die einzelnen Funktionen (Interpolationsparameter) eindeutig auf die beiden Bussysteme übertragen lassen. Das heißt die einzelnen Funktionen müssen sich auf beide Bussysteme so abbilden lassen, dass sich jeweils dasselbe Weg-/Zeitprofil bei der Interpolation für einen Antrieb ergibt.

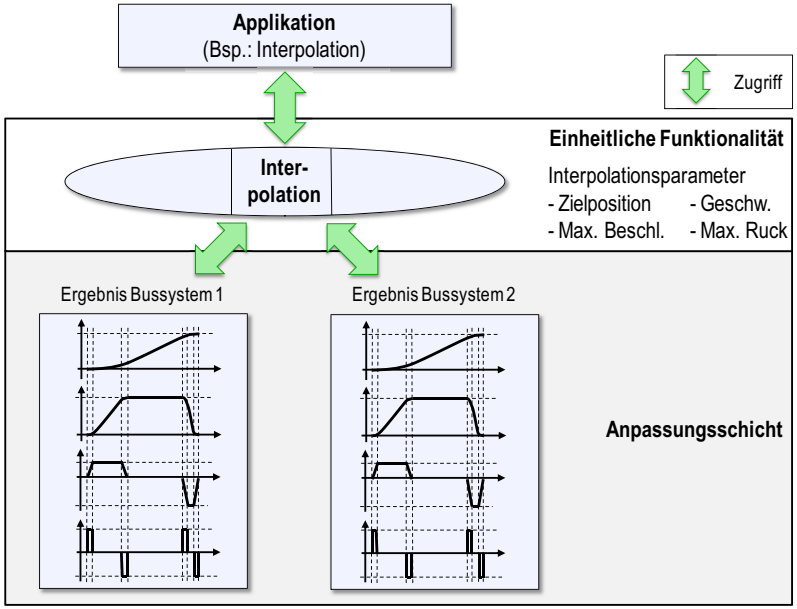


Bild 2.2: Eindeutigkeit der Funktionalitäten und Funktionen am Beispiel der Interpolation

2.2.4 Erweiterbarkeit und Durchgriff auf busspezifische Funktionen

Verfügt ein Bussystem oder Applikationsprofil als einziges über eine Funktionalität oder Funktion, ist es aufgrund der Anzahl der vereinheitlichten Funktionen nicht sinnvoll, diese in eine vereinheitlichte Schnittstelle mit aufzunehmen. Um jedoch sowohl der Bussystemunabhängigkeit als auch der Vollständigkeit gerecht zu werden, muss trotzdem darauf zugegriffen werden können. Dazu soll eine Möglichkeit geschaffen werden, über die einheitliche Schnittstelle auf busspezifische Funktionen zugreifen können. Damit ist auch die Erweiterbarkeit hinsichtlich zukünftiger, bisher nicht bekannter Funktionalitäten gegeben.

2.2.5 Abbildung auf herstellerspezifische Funktionalitäten

Sind einzelne Funktionalitäten der einheitlichen Schnittstelle nicht durch die Spezifikation des Bussystems oder Applikationsprofils festgelegt, soll es möglich sein, diese Funktionalitäten der Applikationsschnittstelle auf herstellerabhängige Funktionalitäten abbilden zu können (Bild 2.3).

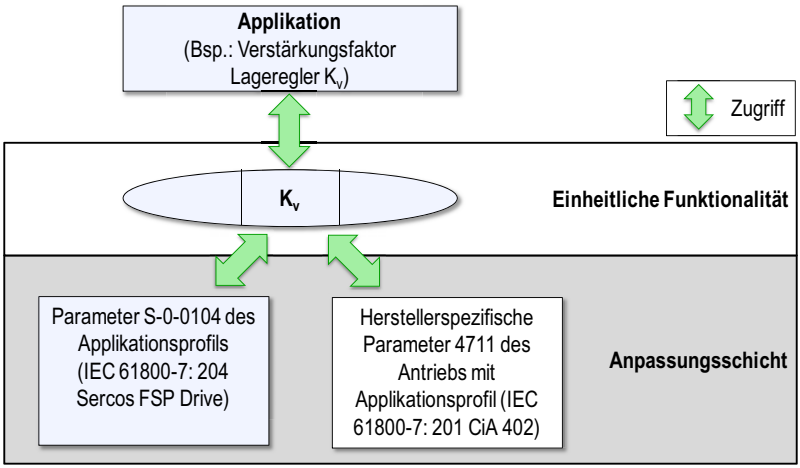


Bild 2.3: Abbildung auf herstellerspezifische Funktionalitäten

Im abgebildeten Beispiel kann der Verstärkungsfaktor der einheitlichen Schnittstelle auf den Parameter S-0-0104 des Sercos-Antriebsprofils (FSP Drive) abgebildet werden. Das Profil CiA 402 unterstützt diesen Parameter nicht. Er kann jedoch auf den herstellerspezifischen Parameter eines Antriebs abgebildet werden und steht somit für diesen Antrieb der Applikation über die funktional einheitliche Schnittstelle zur Verfügung.

2.2.6 Echtzeitfähigkeit der Schnittstelle

Echtzeitfähigkeit
Die Applikationsschnittstelle muss echtzeitfähig sein und Mechanismen zur Verfügung stellen, um die Applikation mit der Kommunikation zu synchronisieren. Sie muss unabhängig von der Umsetzung der Echtzeitfähigkeit des jeweiligen Bussystems eine einheitliche Möglichkeit des Austauschs der Echtzeitdaten bieten.

Konfigurierbarkeit der Echtzeitdaten
Die Schnittstelle muss bezüglich ihrer Echtzeitdaten konfigurierbar sein. Jede Applikation benötigt andere Echtzeitdaten von unterschiedlichen Teilnehmern. Diese müssen während der Initialisierung in die Applikationsschnittstelle integriert werden können.

2.3 Zusammenfassung

In diesem Kapitel sind für die vorliegende Arbeit relevante Begriffe definiert und Anforderungen an eine funktional einheitliche Applikationsschnittstelle aufgestellt. Diese Anforderungen sollen von der zu definierenden Schnittstelle für den Zugriff von der Applikation auf das Bussystem, sowie die jeweiligen Applikationsprofile erfüllt werden. Anhand der Anforderungen wird im folgenden Kapitel der Stand der Technik näher untersucht.

3 Stand der Technik und Zielsetzung

In diesem Kapitel werden zunächst anhand des aktuellen Stands der Technik die heutigen industriellen Bussysteme in der Fertigungstechnik erläutert. Zudem werden bisherige Vergleiche und Vereinheitlichungen von Bussystemen dargestellt und hinsichtlich der definierten Anforderungen bewertet. Zusätzlich werden Systemanalyseverfahren aus der Softwaretechnik verglichen und für die Eignung zur Konzeption einer funktional einheitlichen Applikationsschnittstelle bewertet. Mit Hilfe der Ergebnisse wird dann die Zielsetzung für die in dieser Arbeit zu definierende Schnittstelle abgeleitet und das Vorgehen zum Erreichen der Ziele kurz dargestellt.

3.1 Beschreibung von Bussystemen

3.1.1 OSI Schichtenmodell

Bussysteme lassen sich wie andere Kommunikationssysteme ebenfalls in die Schichten des OSI (Open Systems Interconnection)-Referenzmodells /35/ einteilen. Das OSI-Referenzmodell teilt die Kommunikation in sieben Ebenen (Schichten) mit festgelegtem Funktionsumfang ein.

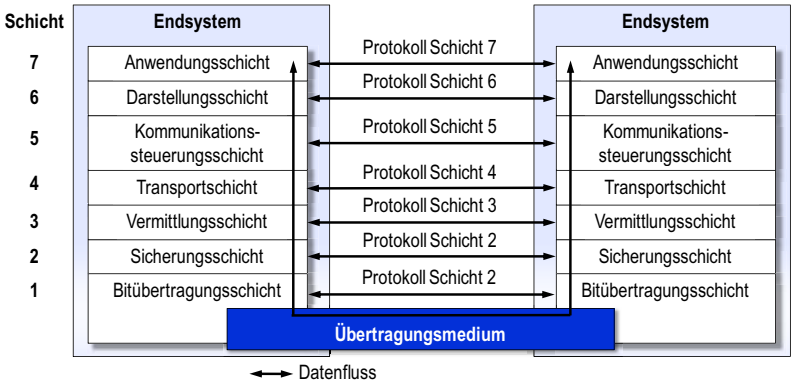


Bild 3.1: Struktur des OSI-Schichtenmodells (Quelle /35/)

Jede Schicht erledigt dabei eine klar definierte Gruppe von Teilaufgaben in der Kommunikation /36/. An definierten Schnittstellen stellt jede Schicht Dienste bereit, die von den Nachbarschichten in Anspruch genommen werden können. Die Schichteneinteilung dient der Abstraktion der Kommunikationsprozesse. Die Aufgliederung der Kommunikation in sieben Ebenen ist in Bild 3.1 dargestellt.

Schicht 1 ist die physikalische Schicht, auch Bitübertragungsschicht genannt. Sie bestimmt, in welcher Weise die Datenübertragung physikalisch zu erfolgen hat, d.h. die elektrischen und mechanischen Eigenschaften der Übertragung. In ihr wird vereinbart, wie die Übertragung der einzelnen Bits erfolgt. Dazu gehören die Art der Modulation, der Spannungspegel für die Übertragung, die vereinbarte Zeitdauer für ein einzelnes Bit, die Wahl der Übertragungsleitung, die Endsystemkopplung (Stecker) und die Zuordnung der Anschlüsse (Pinbelegung) für die Übertragung des Bitstroms. Diese Schicht wird bei Ethernet-basierten Bussystemen von der Ethernet Norm /15/ festgelegt.

Schicht 2 ist die Sicherungsschicht der Leitungsebene. Ihre Aufgabe ist der sichere Transport der Daten von einer Station zu einer anderen Station. Sie dient damit der Datensicherung während der physikalischen Übertragung. Die Daten werden so verpackt, dass Übertragungsfehler von den teilnehmenden Stationen erkannt werden können. Dazu werden die zu übertragenden Daten in Rahmen eingeteilt, so dass in jedem Rahmen immer eine maximale Anzahl von Bytes enthalten ist. Rahmengrößen im Bereich von einigen hundert Bytes sind üblich. Die Rahmen enthalten außer den Rohdaten zusätzliche Informationen für die Übertragung, die die Sicherungsschicht ihrerseits zu den bereits vorhandenen Daten hinzufügt. Mit den hierbei verwendeten Mechanismen soll festgestellt werden, ob Rahmen fehlerhaft übertragen wurden oder ob Rahmen auf dem Übertragungsweg verloren gingen. Zusätzlich ist Schicht 2 für den Buszugriff zuständig. Beim Buszugriff weichen einige der Ethernet-basierten Bussysteme von der Ethernet-Spezifikation ab. Sie verwenden eigene, nicht für Ethernet spezifizierte Buszuteilungsverfahren, wie beispielsweise das Zeitschlitzverfahren.

Die Schichten 3-6 werden bei Ethernet-basierten Bussystemen nicht verwendet und werden in dieser Arbeit deshalb nicht weiter beschrieben.

Schicht 7 ist die Anwendungsschicht. Sie stellt Funktionalitäten für die Anwendungen zur Verfügung. Diese Schicht stellt die Verbindung zu den unteren Schichten her. Auf

diese Ebene greift die Applikation zu. In dieser Schicht sind die gesamten Funktionalitäten eines Ethernet-basierten Bussystems und Applikationsprofils abgebildet.

Ethernet-basierte Bussystem haben meist nur Ausprägungen auf den Schichten 1, 2 und 7 (Bild 3.2) /16/. Allerdings sind die Schnittstellen zwischen diesen Schichten bei Ethernet-basierten Bussystemen nicht exakt festlegt. Es existiert keine klare Trennung zwischen den Schichten. Dies wird insbesondere deutlich, dass die Beschreibung der gesamten Funktionalitäten der Schicht 7 zugeordnet ist. Lediglich die Erweiterungen zur Echtzeit gegenüber Ethernet sind den Schichten 1-2 zugeordnet. Die einzelnen Schichten können auch nicht zwischen unterschiedlichen Bussystemen ausgetauscht oder darauf zugegriffen werden, da sie inkompatibel zueinander sind. Es existiert insbesondere keine Vereinheitlichung hin zur Applikation oberhalb von Schicht 7 /37, 38, 39/.

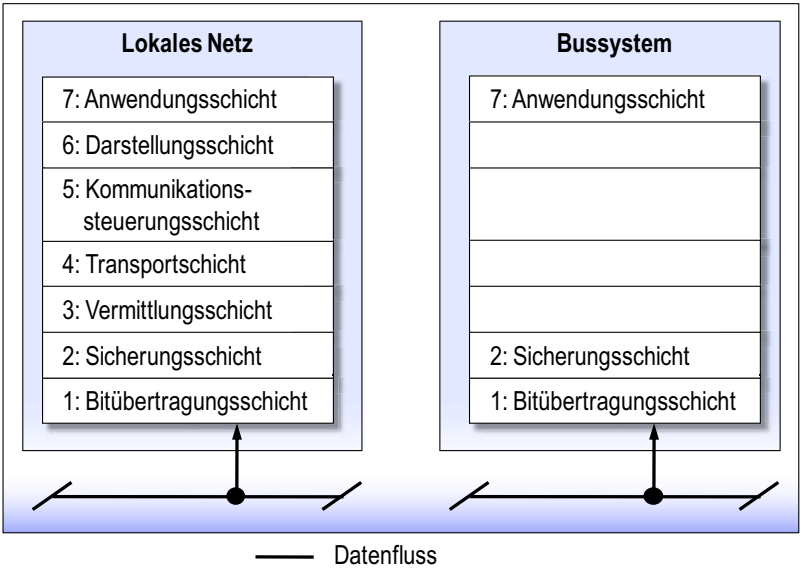


Bild 3.2: OSI-Schichten von Ethernet-basierten Bussystemen (Quelle /6/)

3.1.2 Normung Ethernet-basierter Bussysteme

Die in Europa verbreiteten Ethernet-basierten Bussysteme sind alle in der IEC 61784-2 /40/ genormt. Sie werden in CPFs (Communication Profile Families) eingeteilt. Die Beschreibung der Funktionalitäten ist nicht direkt in der IEC 61784-2 enthalten. Die Familien verweisen auf die jeweils relevanten Teile der IEC 61158 Teil 2 bis 6 /41, 42, 43, 44, 45/. Die Teile der IEC 61158 sind nach den jeweiligen CPFs unterteilt und beschreiben die einzelnen Funktionalitäten und Funktionen des jeweiligen Bussystems. Durch diese beiden Normen und deren Beziehungen zueinander entsteht die Spezifikation für die einzelnen Ethernet-basierten Bussysteme. Sie sind in Tabelle 3.1 nach Ihrer Bezeichnung und der CPF aufgelistet.

Der Ethertype beschreibt das Feld „Typ“ im Ethernet-Telegramm. Mit Hilfe dieses Feldes kann die Netzwerkschnittstelle den Protokolltyp und somit die Struktur des Inhalts im Ethernet-Telegramm zuordnen.

Ethernet-basierte Bussystem und Kommunikationsprofile der IEC 61784		
IEC 61784 Profile	**Bezeichnung**	**Ethertypes**
CPF-2	ControlNet (Ethernet/IP)	(0x800 IP)
CPF-3	PROFIBUS/PROFINET	0x8892
CPF-4	P-Net	(0x0800 IP)
CPF-10	Vnet/IP	(0x0800 IP)
CPF-11	TCnet	0x888B
CPF-12	EtherCAT	0x88A4
CPF-13	Ethernet POWERLINK (EPL)	0x88AB
CPF-14	EPA	0x88BC
CPF-15	MODBUS - RTPS	(0x0800 IP)
CPF-16	SERCOS	0x88CD

Tabelle 3.1: Ethernet-basierte Bussysteme der IEC 61784-2

Bei der Struktur der Norm (Tabelle 3.2) wird ebenfalls deutlich, dass diese nur zwischen den Schichten 1,2 und 7 unterscheidet /52, 46/. Die strukturelle Beschreibung der einzelnen CPF ist nicht vorgeschrieben. Die Normen beinhalten die komplette Beschrei-

bung der Funktionalitäten jedes einzelnen Bussystems. Sie ist jedoch für jedes Bussystem strukturell unterschiedlich und kann somit nicht direkt für eine Vereinheitlichung genutzt werden. Die aus Applikationssicht maßgebliche Beschreibung der Funktionalitäten und Funktionen sind in der Applikationsschicht unstrukturiert beschrieben.

Normteil	Inhalt
IEC 61158-1	Einleitung
IEC 61158-2	Bitübertragungsschicht
IEC 61158-3	Sicherungsschicht Dienste
IEC 61158-4	Sicherungsschicht Protokolle
IEC 61158-5	Anwendungsschicht Dienste
IEC 61158-6	Anwendungsschicht Protokolle

Tabelle 3.2: Struktur der IEC 61158

3.2 Genormte Applikationsprofile

Neben den Normen der reinen Kommunikation der Bussysteme sind Beschreibungen einzelner Geräteklassen als Applikationsprofile spezifiziert. Diese beschreiben Funktionen und Funktionalitäten zur Ansteuerung der jeweiligen Geräteklassen. Sie sorgen für eine Interoperabilität zwischen Geräten unterschiedlicher Hersteller einer Geräteklasse /47/.

Für digitale Antriebe existieren vier spezielle Applikationsprofile /48/. Diese sind in der IEC 61800-7 /49/ genormt. Die Norm beschreibt die unterschiedlichen Antriebsprofile

- CiA 402,
- CIP Motion,
- Profidrive und
- Sercos FSP Drive.

Das CiA 402 (CAN in Automation) Profil ist hauptsächlich für das Feldbussystem CANopen /50/, das auf CAN (Controller Area Network) /51/ basiert, entwickelt worden. CIP (Common Industrial Protocol) Motion wurde für Ethernet/IP spezifiziert. Profidrive

wurde für Profibus und Profinet ausgelegt und das Sercos FSP (Function Specific Profile) Drive entstand speziell für Sercos I-III.

Die IEC 61800-7 hatte ursprünglich das Ziel, für eine Vereinheitlichung dieser vier Antriebsprofile zu sorgen. Die einheitliche Schnittstellenbeschreibung PDS (Power Drive System) sollte einen einheitlichen Zugriff auf die vier einzelnen Antriebsprofile beschreiben (Bild 3.3). Hierbei handelt es sich um eine sehr abstrakte Sicht auf elektrische Antriebe. Diese Sicht erlaubt es nicht, aufgrund des hohen Abstraktionsgrades, eine funktionale Schnittstelle abzuleiten, die den Zugriff aus einer Applikation auf einen Antrieb vereinheitlicht. Diese abstrakte Sicht wird lediglich dazu verwendet, um die einzelnen spezifischen Antriebsprofile grob zu strukturieren. Auf Grund einer fehlenden Einigung wurden die Profile im Einzelnen ohne jeglichen Bezug zueinander beschrieben /52, 53/.

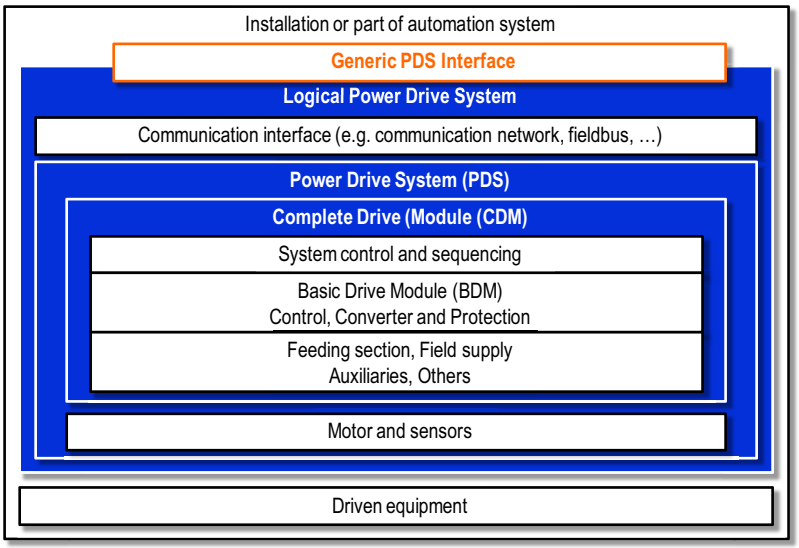


Bild 3.3: Architektur der Spezifikation für elektrische Antriebe
(Quelle: IEC 61800-7-1 /49/)

Es wird in der IEC 61800-7 lediglich eine Abbildung (Mapping) einzelner Antriebsprofile auf Bussysteme beschrieben, die keine eigenen Applikationsprofile spezifizieren. Beispielsweise wird das Mapping des Sercos FSP Drive und CiA 402 auf das Bussystem Ethercat beschrieben (Bild 3.4).

Dieses abstrakte Modell der IEC 61800-7 bietet keine ausreichende Grundlage für eine Vereinheitlichung von Ethernet-basierten Bussystemen und deren Applikationsprofile. Der Grund hierfür ist, dass lediglich einzelne Applikationsprofile parallel in unterschiedlicher Struktur beschrieben werden.

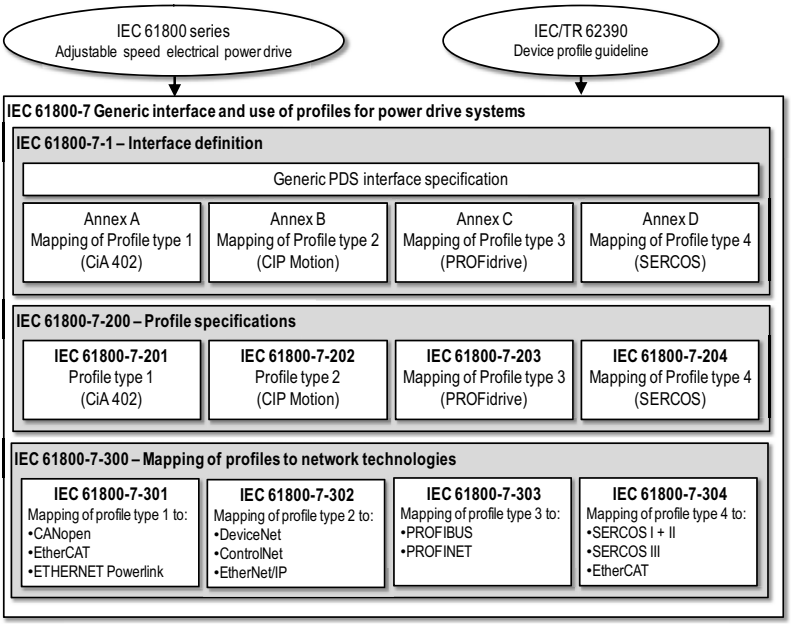


Bild 3.4: Struktureller Aufbau der IEC 61800-7 (Quelle: IEC 61800-7 /49/)

3.3 Vergleich von Bussystemen

Bisherige Vergleiche von Bussystemen erfolgen anhand von Kennzahlen zur Leistungsfähigkeit, wie beispielsweise der Echtzeitfähigkeit, dem Jitter und der Anzahl adressier-

barer Teilnehmer /54, 55, 56, 57/. Die Vergleiche erlauben einen objektiven Vergleich, da sich diese Zahlen einfach belegen lassen. Da die Leistungsfähigkeit der einzelnen Bussysteme weit über dem liegt, was in den meisten Applikationen benötigt wird, sind diese Kennzahlen eher für Marketingaspekte zu nutzen. Sie liefern keine Ergebnisse, welche funktionalen Anforderungen von Applikationen an Bussysteme gestellt werden.

Werden die Funktionalitäten der Bussysteme sowie deren Umsetzung und Zugriffsmöglichkeiten aus Applikationssicht betrachtet, handelt es sich um eine Beschreibung und einen Vergleich der Leistungsfähigkeit einzelner Funktionalitäten /58, 59/.

Die existierenden Vergleiche und Beschreibungen dienen der Erklärung der einzelnen Funktionalitäten, aber eignen sich nicht als Grundlage für eine funktionale Vereinheitlichung solcher Bussysteme.

3.4 Einheitliche Bussystemschnittstellen

3.4.1 Einheitliche Softwareschnittstellen

Bisherige Arbeiten beschäftigen sich mit dem Thema der einheitlichen Bussystemschnittstelle und generellen Austauschmechanismen /60, 61/. Hierbei werden Austauschmechanismen wie Dual-Ported-Ram und Interrupthandling zur Synchronisation betrachtet. Die eigentlichen Funktionalitäten, die von der Applikation genutzt werden, werden hierbei nicht berücksichtigt. Somit bleibt immer noch busspezifisches Wissen beim Zugriff auf Funktionen und Funktionalitäten der einzelnen Systeme auf beiden Seiten der Schnittstellen bestehen.

In /62/ wird eine einheitliche Applikationsschnittstelle für Feldbusse definiert. Die definierten Dienste sind heute bereits weitestgehend durch Ethernet bei den Ethernet-basierten Bussystemen abgedeckt und daher nicht mehr relevant. Applikationsprofile waren zum damaligen Zeitpunkt nicht existent und somit auch nicht in der Betrachtung der Applikationsschnittstelle enthalten.

Die Arbeiten /63, 64/ beschreiben ein Klassenkonzept für Bussystemobjekte. Es wird eine Softwarearchitektur definiert, in der eine Hierarchie, ausgehend von einer allgemeingültigen Klasse bis hin zu einer herstellerspezifischen Abbildung (Bild 3.5), enthalten ist. Diese Struktur dient jedoch nicht einer funktional einheitlichen Betrachtung von

Buskomponenten aus Applikationssicht. Diese Lösung beschreibt lediglich eine Strukturierung von Geräteklassen.

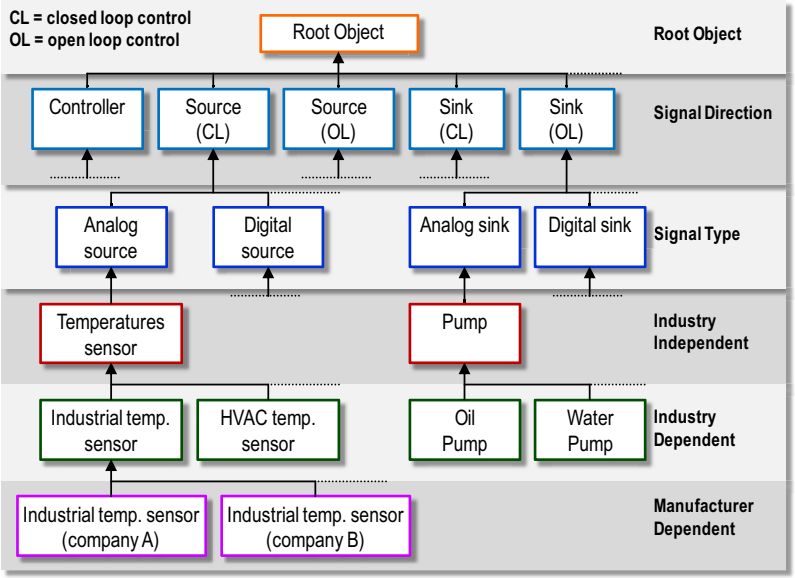


Bild 3.5: Profilhierarchie für ein branchenübergreifendes Klassenkonzept für Bussystemobjekte (Quelle /63/)

3.4.2 Einheitliche Hardwarearchitektur

In der Arbeit /65/ wird ein optimiertes Bussystem beschrieben. Bei der Optimierung werden Buszugriff und Echtzeitverhalten betrachtet. Eine Vereinheitlichung zur Applikation hin wird nicht durchgeführt. Diese Lösung stellt somit lediglich eine weitere mögliche Beschreibung und Umsetzung auf den Schichten 1 und 2 des OSI-Schichtenmodells dar.

Auch Chiphersteller, die Produkte anbieten, die mehrere Bussysteme unterstützen /66/, benötigen auf Seiten der Applikation busspezifisches Wissen zum Bedienen der Chip-Schnittstellen. Das busspezifische Verhalten des Chips wird durch eine entsprechende

Firmware definiert, die für jedes Bussystem unterschiedlich ist. Der Inhalt der Speicherschnittstelle ist abhängig vom Bussystem. Der Chip übernimmt lediglich die Kommunikation auf den Ebenen 1-2 des OSI-Schichtenmodells. Der Chip beinhaltet die busspezifischen Ausprägungen der Chips für MAC (Media Access Control) und PHY (physikalische Schnittstelle zum Netzwerk) zur busspezifischen Umsetzung der Echtzeitfähigkeit (Bild 3.6).

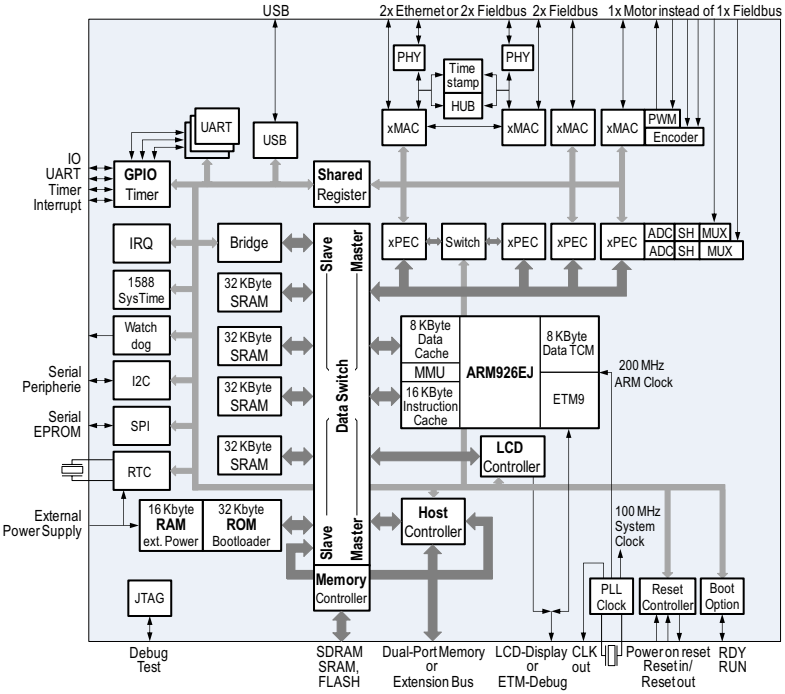


Bild 3.6: Architektur des NetX Chips der Firma Hilscher (Quelle /66/)

Ebenfalls Schnittstellen zu Busadaptern wie Anybus /67/ sind nicht durchgängig vereinheitlicht. Diese Schnittstellen verfügen über eine definierte Hardwareschnittstelle, benötigen aber auf Seiten der Applikation trotzdem busspezifisches Wissen. Die Schnittstelle mit einer Mailbox regelt lediglich den azyklischen Datenaustausch, macht aber keine einheitliche Festlegung über den Inhalt der Daten und bleibt somit busspezifisch. Um

ganze Protokolle zu übersetzen, wird keine einheitliche Schnittstelle genutzt, sondern es wird jeder Parameter eines Bussystems einzeln auf den eines anderen Bussystems übersetzt.

All diese Ansätze vereinen lediglich die benötigten hardwaretechnischen Komponenten der Schichten 1 und 2 des OSI-Modells in einem Chip oder einer Elektronik und können deren Verhalten auf das jeweilige Bussystem anpassen. Sie stellen eine sinnvolle Ergänzung zur vorliegenden Arbeit dar, da sie auf Hardwareebene eine Anpassung an die jeweiligen Bussysteme ermöglichen. Sie übernehmen einen Großteil der spezifischen Kommunikation, wie Buszugriff und Telegrammaufbau. Sie stellen dieses jedoch nicht vereinheitlicht der Applikation zur Verfügung.

3.5 Echtzeitmiddleware

Middleware ist eine Schicht oberhalb des Betriebssystems in einem verteilten System /6/. Sie dient zur Vernetzung verteilter Anwendungen unabhängig von der Rechenhardware, Betriebssystem und Kommunikation (Bild 3.7). Eine Middleware kapselt die gesamten Schichten, die zwischen zwei Anwendungen liegen. Sie erlauben es, gesamte Objektstrukturen zwischen Anwendungen auszutauschen, als wären sie direkt miteinander gekoppelt.

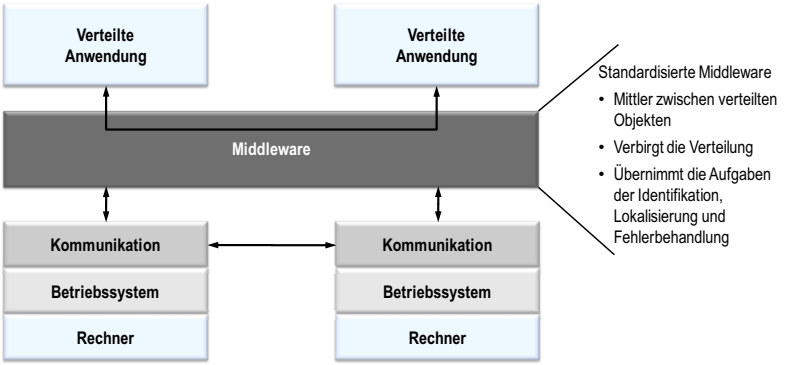


Bild 3.7: Architektur einer verteilten Anwendung über Middleware (Quelle /6/)

Spezielle Umsetzungen von Middleware über Ethernet sind um Quality of Services (QoS) ergänzt. Diese Ergänzungen machen die Middleware echtzeitfähig, indem sie eine Priorisierung gegenüber anderen Protokollen bei der Kommunikation erlauben. Beispiele für Umsetzungen hierfür sind:

- RT-CORBA (Real Time- Common Object Request Broker Architecture)
- OSA+ (Open System Architecture – Platform for Universal Services)
- DCOM (Distributed Component Object Model)

Middleware löst die Interaktion, Lokalisierung und Identifikation zwischen verteilten Anwendungen. Die Interaktion erlaubt den Datenaustausch und mit der Lokalisierung und Identifikation lassen sich Kommunikationsteilnehmer auffinden und deren Funktionsumfang genau ermitteln. Eine einheitliche Beschreibung der Dateninhalte aus Applikationssicht existiert jedoch nicht. Zusätzlich wird vorausgesetzt, dass alle der verteilten Anwendungen, die Middleware unterstützen, und die Protokolle über das unterlagerte Kommunikationssystem übertragen werden können.

Auch die konsequente Verwendung der QoS von Standard-Ethernet als einheitliches Bussystem wird ebenfalls als Grundlage für eine einheitliche Kommunikationslösung betrachtet. In /68/ werden die Funktionalitäten für eine topologieabhängige Adressierung, einen optimierten Telegrammaufbau und die Sicherstellung der Synchronität als grundlegende Kommunikationslösung beschrieben.

Für die industrielle Kommunikation und den darauf aufsetzenden Applikationen stellen Middleware und Standard-Ethernet mit QoS jedoch nur weitere Kommunikationssysteme dar, die über eines der vorhandenen Profile aus der Applikation heraus angesprochen werden können. Sie erreichen aufgrund der zusätzlich zu übertragenden Daten nicht die Leistungsfähigkeit von Ethernet-basierten Bussystemen. Eine funktionale Vereinheitlichung wird ebenfalls nicht erreicht.

3.6 Offene Steuerungsarchitekturen

Arbeiten im Bereich von offenen Steuerungsarchitekturen /69, 70/ wie OSACA (Open System Architecture for Controls Within Automation Systems) /71/ haben sich zum Ziel gesetzt, ein rekonfigurierbares Komponentensystem zur Erstellung von Steuerungssoftware zu realisieren (Bild 3.8). Dabei wird eine Plattform beschrieben, die es erlaubt, AO

(Applikationsobjekte) auszutauschen. Applikationsobjekte sind beispielsweise Interpreter oder Interpolator.

Eine ähnliche Architektur namens SOFIA (modulares Softwaresystem für intelligente Antriebe) beschreibt eine Plattform für eine modulare Antriebsarchitektur /72/ (Bild 3.9). Der Nutzen, der sich aus der Modularisierung ergibt, besteht darin, dass Komponenten (K) unabhängig voneinander entwickelt und leicht wiederverwendet werden können. Die Kommunikation zwischen den Komponenten erfolgt über eine Middleware, das SOFIA Component Interface (SCI). Die Konfiguration des Datenaustauschs zwischen den Komponenten wird in der SOFIA Communication Description (SCD) beschrieben.

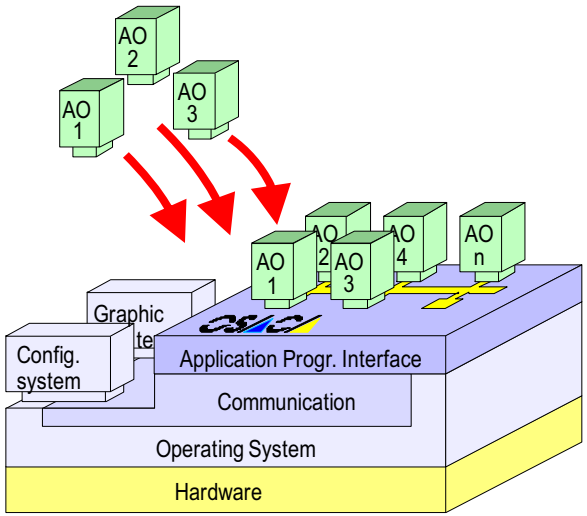


Bild 3.8: Architektur der OSACA Plattform (Quelle /69/)

Diese Middleware vereinheitlicht nicht den Zugriff auf das Bussystem, sondern rein den Zugriff auf Softwarekomponenten und deren Datenaustausch innerhalb eines Antriebs, bzw. einer Steuerung bei OSACA. Der Wechsel auf ein anderes Bussystem erfordert dennoch Änderungen an den Applikationsobjekten, da die Zugriffe auf das Bussystem nicht vereinheitlicht werden. Es werden lediglich Schnittstellen zum Datenaustausch zwischen den einzelnen Modulen definiert.

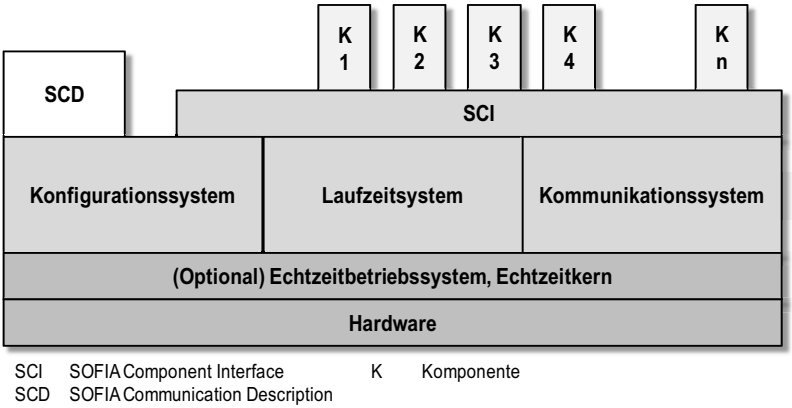


Bild 3.9: Architektur der SOFIA-Plattform (Quelle /72/)

3.7 FDT/DTM

Die Spezifikationen FDT/DTM (Field Device Tool/ Device Type Manager) /73, 74/ und EDDL (Electronic Device Description Language) /75/ bieten ein Framework bzw. eine Beschreibungsform zur Konfiguration von Busteilnehmern. Sie beschreiben die bus- und herstellerspezifischen Darstellungsformen innerhalb eines Konfigurators /76/. Ein solcher Konfigurator erhält über die DTMs Parameter und ActivX-Objekte /77/, die er dem Nutzer darstellt. Diese DTMs werden über die FDT-Schnittstelle in den Konfigurator eingebunden (Bild 3.10). Die Kommunikation zum Bussystem ist in den DTMs nicht vereinheitlicht. Lediglich die Schnittstelle zum FDT und zur Darstellung in einem Konfigurator ist vereinheitlicht. Das FDT selbst muss für jedes Bussystem angepasst werden und dient nicht zum Austausch von Echtzeitdaten.

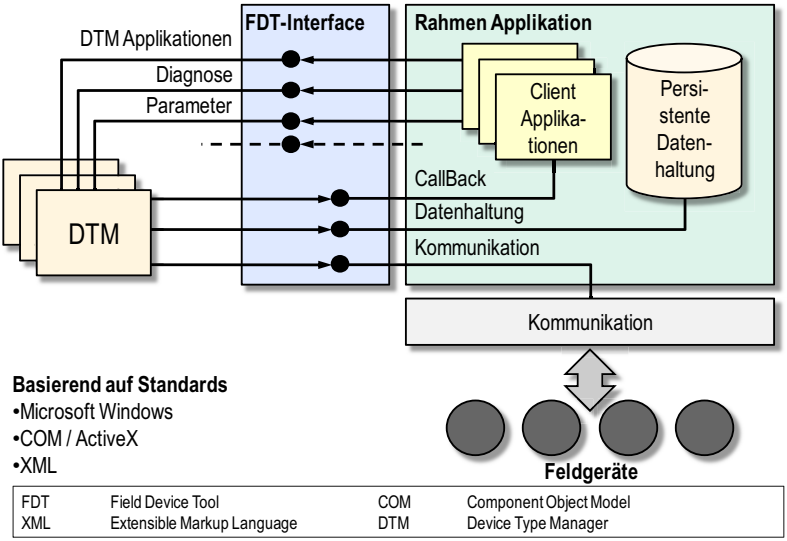


Bild 3.10: Architektur von FDT/DTM (Quelle /73/)

3.8 PLCopen

Die PLCopen /78/ ist eine hersteller- und produktunabhängige internationale Vereinigung und wurde 1992 gegründet. Ursprünglich wurde durch sie die Normung von Programmiermethoden in der IEC 61131-3 vorangetrieben. Heutige Aufgaben bestehen darin, Bibliotheken für häufig auftretende Aufgabenstellungen in der Automatisierung zu definieren. Es werden Funktionsbausteine (FB) definiert, die in der SPS nicht programmiert, sondern lediglich verknüpft und parametriert werden müssen. Diese Definition findet in den Technical Committees (TC) statt. Es gibt TCs für:

- Programmiersprachen
- Schulung
- Motion Control
- Zertifizierung
- Kommunikation zur Leitebene
- Sicherheitstechnik
- XML

Mit Hilfe der Funktionsbausteine wird eine Applikation konfiguriert, indem mehrere einzelne Funktionsbausteine zu Programmen und schließlich zu einer gesamten Applikation kombiniert werden (Bild 3.11).

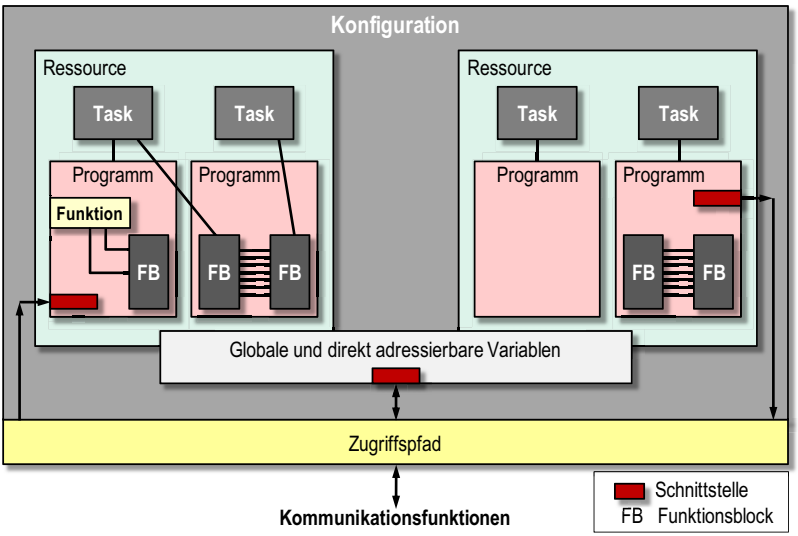


Bild 3.11: Konfiguration von PLCopen-Bausteinen zu einem Programm (Quelle /78/)

Diese Funktionsbausteine sind jedoch abhängig von der Funktionalität und vereinheitlichen nicht den Zugriff auf unterschiedliche Bussysteme und Applikationsprofile. Für jedes Bussystem und Applikationsprofil müssen die Funktionsbausteine entsprechend angepasst werden. Bisher existieren nur Definitionen für den Bereich Motion Control und Sicherheitstechnik.

3.9 Defizite

Keiner der aufgeführten wissenschaftlichen Ansätze bzw. industriellen Umsetzungen erfüllt die Anforderung an eine bussystemunabhängige Schnittstelle zwischen der Applikation und der Kommunikation vollständig. Die Defizite der einzelnen Ansätze hinsichtlich der Anforderungen aus Kapitel 2 sind in Tabelle 3.3 zusammengefasst dargestellt. Dabei wird deutlich, dass insbesondere keine Ansätze existieren, die ohne bussystem-

spezifische Informationen auskommen. Es sind beim Wechsel der Applikation auf ein anderes Bussystem Anpassungen notwendig. Die Recherche im Stand der Technik zeigt, dass ebenfalls keine strukturierte Sichtweise auf Ethernet-basierte Bussysteme existiert, da bei bisherigen Entwicklungen und Normungen mehrere Bussysteme meist parallel liefen.

Anforderung		Beschreibungsformen				Umsetzungen					
		OSI Schichtenmodell	IEC 61784-2/61185	IEC 61800-7	Bisherige Vergleiche	Bisherige Softwareschnittstellen	Bisherige Hardwareschnittstelle	Middleware	Offene Steuerungsarchitektur	FDT/DTM	PLCopen
Universalität	Applikationsunabhängigkeit	●	●	○	○	●	●	●	●	●	○
	Geräteunabhängigkeit	●	●	○	●	○	●	●	●	○	●
	Bussystemunabhängigkeit	○	○	○	○	○	○	○	○	○	○
	Hardwareunabhängigkeit	●	●	●	●	●	○	●	○	●	●
	Vollständigkeit	●	●	●	○	○	●	●	◗	○	○
Handhabbarkeit	Strukturierung	◗	○	○	○	○	○	○	○	○	○
	Eindeutigkeit	○	●	●	○	○	●	○	●	○	●
Erweiterbarkeit	Busspezifisch	●	●	●	◗	○	●	●	●	●	○
	Hersteller	●	●	●	○	○	●	●	●	○	●
Echtzeitfähigkeit	Echtzeitfähigkeit	●	●	○	○	●	●	◗	●	○	●
	Konfigurierbarkeit	○	○	○	○	◗	○	●	●	●	○

Tabelle 3.3: Übersicht der Anforderungen und dem Stand der Technik

Als Grundlage für diese Arbeit können die Spezifikationen für Ethernet-basierte Bussysteme und Applikationsprofile verwendet werden, da sie die komplette Beschreibung der Funktionalitäten jedes einzelnen Systems beinhalten. Sie beschreiben die einzelnen Systeme sehr detailliert, jedoch in unterschiedlicher Struktur, anhand derer sich keine direkte Vereinheitlichung ableiten lässt.

Konzepte zu standardisierten Schnittstellen und Applikationsmodulen wie Middleware, offene Steuerungsarchitekturen und PLCopen-Bausteine stellen jedoch eine sinnvolle Ergänzung zu dieser Arbeit dar, da sie eine modulare Applikationsentwicklung unterstützen. Sie verallgemeinern den Zugriff auf einer höheren Ebene innerhalb der Applikation und benötigen bisher noch eine jeweils busspezifische Anpassung beim Wechsel des darunter liegenden Kommunikationssystems.

3.10 Systemanalyseverfahren der Softwaretechnik

Um den Defiziten aus dem Stand der Technik zu begegnen, wird eine geeignete Methode zur Definition einer funktional einheitlichen Applikationsschnittstelle ermittelt. Dazu werden zwei gängige Systemanalysemethoden aus der Softwaretechnik näher betrachtet, anhand derer die Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme durchgeführt wird.

3.10.1 Strukturierte Analyse

Bei der Strukturierten Analyse /79/ wird ein komplexes Problem in handhabbare Teilbereiche zerlegt, die relativ isoliert voneinander behandelt werden können. Die Zerlegung erfolgt dabei nach unterschiedlichen Gesichtspunkten:

- funktionsorientiert
- datenorientiert

Es wird von einer abstrakten Beschreibung eines Systems oder Vorgangs ausgegangen, dessen Funktionen und Daten im Verlauf der Analyse weiter detailliert werden. Ergebnisse der Strukturierten Analyse sind:

- Datenflussdiagramm
- Datenlexikon
- Primitive Prozessspezifikation

Ausgehend von den Ergebnissen der strukturierten Analyse erfolgt im strukturierten Entwurf die endgültige Konzeption und Realisierung der Software. Zwischen der Analyse und dem Entwurf entsteht jedoch ein Bruch in den Beschreibungssprachen. Sie können nicht in den nächsten Analyseschritt übernommen werden. Dies ist ungeeignet für ein iteratives Vorgehen.

3.10.2 Objektorientierte Analyse

Bei der Objektorientierten Analyse (OOA) werden nicht nur Daten und Funktionen, sondern auch deren Zusammenhänge und Beziehungen beschrieben /80, 81, 82/. Es las-

sen sich folgende, voneinander unabhängige, Abstraktionsarten bei der Objektorientierten Analyse definieren /83/:

- Klassifizierung
  Charakterisierung einer Menge von Individuen (Exemplaren) mit gemeinsamen Attributen durch einen Typ oder eine Klasse.
- Komposition
  Zusammenfassung einer Menge von Individuen mit teilweise gemeinsamen Merkmalen zu einem neuen Individuum mit neuen Merkmalen, welches die Gesamtheit der zusammengefassten Merkmale repräsentiert
- Generalisierung
  Zusammenfassung einer Menge von Individuen mit teilweise gemeinsamen Attributen durch ein übergeordnetes Individuum, welches nur die gemeinsamen Attribute aufweist.
- Benutzung
  Verwendung von Leistungen eines Individuums (oder einer Menge von Individuen) durch ein anderes Individuum zum Zweck der Erbringung eigener Leistungen

Das OOA-Modell wird in ein statisches und dynamisches Teilmodell unterteilt (Bild 3.12). Im statischen Teilmodell werden Assoziationen, Vererbungsbeziehungen zwischen Klassen und Assoziationen untereinander betrachtet. Sie können zu Paketen zusammengefasst werden. Im dynamischen Teilmodell werden mittels Geschäftsprozessdiagrammen, Zustandsautomaten, Szenarien und Botschaften die Interaktionen zwischen Klassen abgebildet. Diese beiden Teilmodelle nutzen elementare Basiskonzepte wie Objekte und Klassen mit ihren jeweiligen Attributen und Operationen.

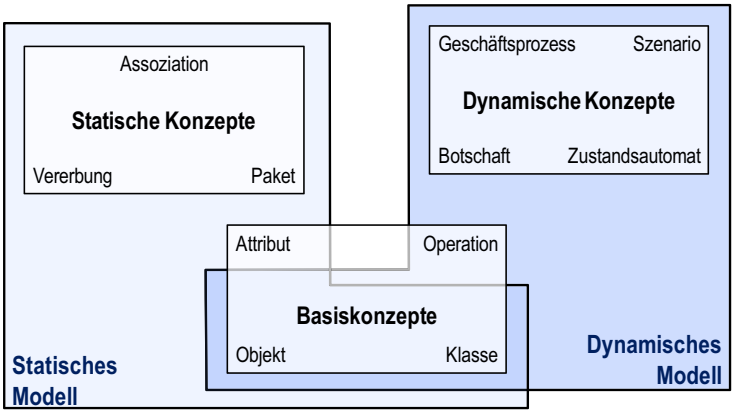


Bild 3.12: Aufteilung eines OOA-Modells in ein statisches und ein dynamisches Teilmodell (Quelle /81/)

Im Entwicklungsprozess folgt nach der Objektorientierten Analyse das Objektorientierte Design (OOD) und das Objektorientierte Programmieren (OOP). Die Beschreibungssprache der Unified Modeling Language (UML) /84, 85, 86/ bleibt über alle Entwicklungsphasen hin gleich. Es findet eine Detaillierung der Beschreibung bis hin zur Implementierung statt.

3.10.3 Gegenüberstellung der Analyseverfahren

In Tabelle 3.4 sind die beiden Analyseverfahren aus der Softwaretechnik gegenübergestellt. Dabei wird deutlich, dass sich die Objektorientierte Analyse am besten eignet, um eine funktional einheitliche Schnittstelle für Ethernet-basierte Bussysteme abzuleiten. Die Objektorientierte Analyse bietet die Möglichkeit, Daten und Operationen gemeinsam zu betrachten. Bei der mechatronischen Betrachtung von Automatisierungsgeräten werden ebenfalls Daten und Funktionen kombiniert betrachtet. Zusätzlich lässt sie eine problembezogene Abstraktion zu, die es erlaubt, die konkrete Lösung zu einem relativ späten Zeitpunkt festzulegen, ohne sich von einer speziellen Implementierung abhängig zu machen.

Die Objektorientierte Analyse bietet eine durchgängige methodische Beschreibung, so dass die Ergebnisse über diese Arbeit hinaus weiter genutzt werden können. Sie können für neue Funktionalitäten und Funktionen erweitert werden. Außerdem lassen sie sich auf weitere Bussysteme und Applikationsprofile anwenden. Sind Änderungen oder Erweiterungen durch zusätzliche Funktionalitäten notwendig, können diese bei der objektorientierten Analyse durch lokale Erweiterungen in die Beschreibung integriert werden.

	Strukturierte Analyse	Objektorientierte Analyse
Orientierung	Mehr technisch orientiert	Entspricht menschlicher Denkweise
Methodenansatz	Trennung von Daten und Operationen	Einheiten aus Daten und Operationen (Klassen)
Abstraktions-möglichkeiten	Modellierung auf den Lösungsbereich bezogen	Modellierung stärker auf den Problembereich bezogen
Methodische Durchgängigkeit	Strukturbruch zwischen Entwicklungsphasen durch unterschiedliche Darstellungen	Die selben Konzepte in allen Entwicklungsphasen
Entwicklungs-unterstützung	Geringere Flexibilität bei Änderungen und Erweiterungen	Evolutionär (schrittweise) Entwicklung wird unterstützt
Stabilität	Bei Anpassungen erhebliche Änderungen des Programmgefüges notwendig	Änderungen eher lokal begrenzt durch Kapselung von Daten und Operationen

Tabelle 3.4: Gegenüberstellung der beiden Analyseverfahren (In Anlehnung an /83/)

3.11 Zielsetzung und Aufgabenstellung

In der Einleitung und dem Stand der Technik wurde ausführlich dargelegt, dass eine Vielzahl Ethernet-basierter Bussysteme existieren, die den Anforderungen zum Datenaustausch von aktuellen und zukünftigen dezentralen Automatisierungskomponenten in der Automatisierungstechnik gerecht werden. Der Datenaustausch zwischen diesen Automatisierungskomponenten in Echtzeit über Ethernet-basierte Bussysteme wird weiter zunehmen. Diese Bussysteme existieren parallel am Markt und unterscheiden sich jedoch in ihrer Leistungsfähigkeit und ihrem Funktionsumfang kaum. Sie sind absolut inkompatibel zueinander und eine Einigung auf ein System ist aus firmenpolitischen Interessen und existierenden Realisierungen ausgeschlossen.

Zu jedem der Ethernet-basierten Bussysteme existieren detaillierte Spezifikationen zu Kommunikation und Applikationsprofilen, die jedoch unterschiedlich strukturiert und detailliert beschrieben sind. Es existiert keine einheitliche Beschreibung der Systeme, die einen direkten funktionalen Vergleich zulassen, aus dem sich eine geeignete Schnittstelle ableiten lässt.

Kann die System-, Applikationsentwicklung und Inbetriebnahme für die Kommunikation über diese Bussysteme vereinfacht werden, bietet dies technische und wirtschaftliche Vorteile.

Technische Vorteile werden dadurch erreicht, dass weniger Varianten in der Softwareentwicklung von Applikationen existieren und damit die Robustheit gegenüber Fehlern bei der Entwicklung erhöht wird. Die Wiederverwendung von Softwareprogrammen in der Applikations- und Systemebene werden, aufgrund der Unabhängigkeit vom eingesetzten Kommunikationssystem, erst durchgängig ermöglicht.

Wirtschaftliche Vorteile ergeben sich durch den reduzierten Entwicklungs-, Inbetriebnahme und Wartungsaufwand. Die Festlegung der Hersteller auf eines oder wenige Bussysteme ist nicht länger notwendig, da eine Anpassung auf ein weiteres Bussystem mit geringem Aufwand und lokalen Änderungen vorgenommen werden kann. Hierbei lassen sich auch neue Märkte erschließen, da die Verbreitung der Bussysteme regional unterschiedlich ist.

Ziel dieser Arbeit ist es, eine funktional einheitliche Applikationsschnittstelle zu definieren. Diese Zielarchitektur (Bild 3.13) soll es erlauben, die Applikation unabhängig von der Kommunikation zu entwickeln. Der Austausch der Kommunikation soll lediglich lokale Änderungen in der Anpassungsschicht erfordern.

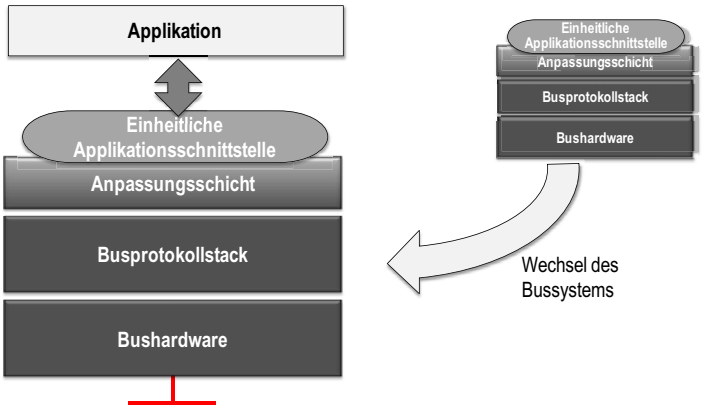


Bild 3.13: Zielarchitektur mit Austausch der Anpassungsschicht beim Wechsel des Bussystems

Dazu sind in dieser Arbeit die notwendigen Grundlagen zu schaffen, die einen funktionalen Vergleich von Ethernet-basierten Bussystemen und Applikationsprofilen erlauben. Die Anforderungen aus Kapitel 2 sind bei der Konzeption der Schnittstelle zu berücksichtigen.

Die einzelnen Bussysteme unterscheiden sich in ihrer Architektur und Sicherstellung des Echtzeitverhaltens /87/. Dabei können drei Gruppen definiert werden, die sich unterscheiden, in welcher Ebene die Echtzeiterweiterung umgesetzt ist (Bild 3.14). Die Echtzeitfähigkeit zur Applikationsebene hin hängt von der Echtzeitfähigkeit der unterlagerten Schichten ab.

Somit ist bei den Systemen der Gruppe (I), die auf TCP/IP (Transmission Control Protocol / Internet Protocol) und UDP (User Datagram Protocol) aufsetzen, nur eine verhältnismäßig große Zykluszeit (>1ms) möglich.

Gruppe II nutzt Standard-Ethernet Hardware und definiert eigene Protokolle zur Erreichung der Echtzeitfähigkeit. Hier können Zykluszeiten bis zu 250µs erreicht werden.

Die performantesten Systeme der Gruppe III, die eine eigene Hardware für den Buszugriff einsetzen, erreichen Zykluszeiten von bis zu 31,25µs.

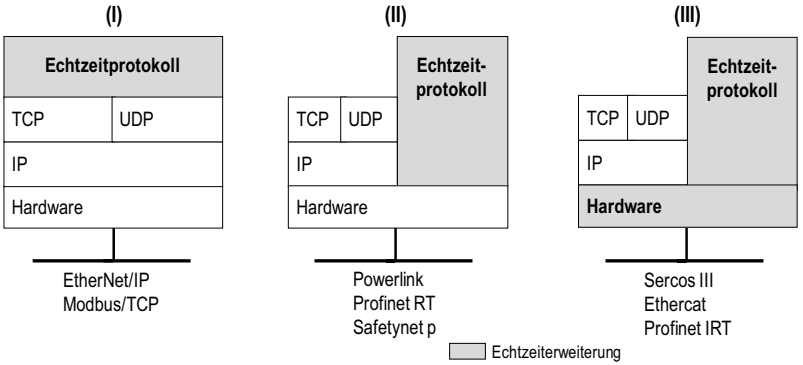


Bild 3.14: Drei Gruppen der Architekturen zur Umsetzung der Echtzeitfähigkeit mit beispielhaften Bussystemen

In dieser Arbeit sollen insbesondere die Systeme der Gruppe III für die Kommunikation betrachtet werden, da diese die aufwändigste Konfiguration der Echtzeitkommunikation erfordern. Die Gruppen I und II können als Untermenge der Gruppe III betrachtet werden.

Bei den Applikationsprofilen werden beispielhaft die Antriebsprofile der IEC 61800-7 /49/ zur Vereinheitlichung herangezogen. Digitale Antriebe stellen bei Automatisierungssystemen neben E/As die wesentlichen Komponenten dar. Profile für digitale und analoge Ein- und Ausgänge sind teilweise mit den Spezifikationen des Bussystems verbunden. Hierbei werden lediglich Speicherbereiche adressiert und über den Bus übertragen. Die Auswertung und die Konfiguration erfolgt durch die Variablenzuordnung innerhalb der SPS und werden deshalb nicht separat betrachtet. Die E/A-Profile CiA 401 und Sercos FSP I/O werden in dieser Arbeit nicht betrachtet, lassen sich jedoch mit denselben Methoden vereinheitlichen.

Im Einzelnen gliedert sich das Vorgehen dieser Arbeit anhand der Objektorientieren Analyse wie folgt:

1. Abstraktion von Bussystemen
   In einem ersten Kapitel wird eine Abstraktion der betrachteten Bussysteme und Applikationsprofile durchgeführt. Dabei ist zu analysieren, welche Funk-

tionalitäten die einzelnen Bussysteme und Applikationsprofile bieten. Sie werden ausgehend von den Bussystemspezifikationen identifiziert und in einem nächsten Schritt generalisiert. Dazu muss eine gemeinsame Abstraktionsebene zwischen den jeweiligen Bussystemen und Applikationsprofilen gefunden werden. Innerhalb dieser gemeinsamen Abstraktionsebene wird ein struktureller Zusammenhang in Form einer Klassifikation der Funktionalitäten abgeleitet. Diese Abstraktion bietet die Grundlage für einen anschließenden Vergleich.

2. Vergleich der Bussysteme
   In diesem Kapitel werden die einzelnen Funktionalitäten anhand einer Spezialisierung näher betrachtet. Die Funktionalitäten werden dazu in Funktionen unterteilt. Ausgehend von den Spezifikationen für Ethernet-basierte Bussysteme werden die Funktionen bei der Spezialisierung detailliert beschrieben. Die detailliert beschriebenen Funktionen werden zwischen den unterschiedlichen Bussystemen verglichen und sowohl Gemeinsamkeiten als auch Unterschiede herausgearbeitet.

3. Ableiten einer funktional einheitlichen Schnittstelle
   Ausgehend von dem vorhergehenden Vergleich soll in diesem Kapitel eine funktional einheitliche Schnittstelle definiert werden und in vorhandene Systemarchitekturen integriert werden. Dazu werden die vergleichbaren Funktionen und Funktionalitäten einheitlich beschrieben und in die Schnittstelle übernommen. Es ist hierbei insbesondere zu entscheiden, welche Funktionen und Funktionalitäten einheitlich beschrieben werden. Zusätzlich zu den funktionalen Inhalten der Schnittstelle müssen die Schnittstelle an sich und deren Konfiguration definiert werden.

4. Realisierung
   In einem abschließenden Kapitel werden die Machbarkeit und die industrielle Umsetzbarkeit der in dieser Arbeit definierten funktional einheitlichen Applikationsschnittstelle gezeigt. Dazu wird beispielhaft ein mechatronisches Modul einer Holzbearbeitungsmaschine über diese Schnittstelle angesprochen. Die dabei gewonnenen Ergebnisse werden vorgestellt und diskutiert.

4 Abstraktion von Bussystemen durch Identifizierung und Generalisierung ihrer Funktionalitäten

In diesem Kapitel wird eine Abstraktion der betrachteten Bussysteme und Applikationsprofile durchgeführt. Hierzu wird zunächst ein geeigneter Abstraktionsgrad für die Strukturdefinition festgelegt. Danach ist zu analysieren, welche Funktionalitäten die einzelnen Bussysteme und Applikationsprofile bieten, aus der eine gemeinsame Struktur für einen späteren Vergleich hergeleitet werden kann. Die Funktionalitäten werden ausgehend von den jeweiligen Bussystemspezifikationen bestimmt.

4.1 Festlegung des Abstraktionsgrads

Bei der Abbildung komplexer realer Systeme als Modell mit Hilfe von formalen Modellierungssprachen stellt sich die Frage nach dem geeigneten Abstraktionsgrad. Für die Festlegung des Abstraktionsgrades bei der Modellierung existieren keine objektiven Kriterien. Bei der Objektorientierten Analyse kann immer nur eine grobe Eingrenzung bzw. eine Einordung zwischen der realen Implementierung und einer maximal abstrakten Basisklasse vorgenommen werden (Bild 4.1).

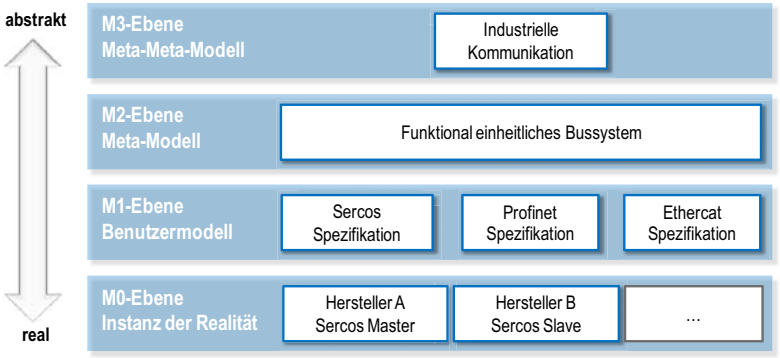


Bild 4.1: Modellierungsebenen der Abstraktion von industriellen Kommunikationssystemen

Ordnet man verfügbare Modellbeschreibungen von Ethernet-basierten Bussystemen den Modellierungsebene (M) der Meta-Modellierung /88, 89/ zu, lassen sich der Modellie-

rungsebene M0 herstellerspezifische Realisierungen in Form von Produkten zuordnen. Sie entsprechen einer konkreten herstellerspezifischen Implementierung, die abhängig vom Bussystem, dem Applikationsprofil und dem Gerät selbst sind.

In der Modellierungsebene M1 sind die jeweiligen Spezifikationen der Kommunikation und Applikationsprofile einzuordnen. Sie stellen eine Abstraktion der herstellerspezifischen Implementierung als Bussystemschnittstelle dar. Sie wird von Nutzerorganisationen, die sich aus Vertretern mehrerer Hersteller zusammensetzen, festgelegt. Die Betrachtung der industriellen Kommunikation auf dieser Ebene abstrahiert die herstellerspezifischen Implementierungsdetails. Es bleiben lediglich busspezifische Modelle übrig. Diese Modellierung wird bei Interoperabilitäts- und Konformitätstests geprüft /90/. Das heißt Komponenten, die diese Tests bestehen, können gegen Komponenten eines anderen Herstellers für dasselbe Bussystem ausgetauscht werden.

Als Grundlage für einen Vergleich und die funktionale Vereinheitlichung der Spezifikationen der Modellierungsebene M1, muss eine geeignete Meta-Modellierung innerhalb der Modellierungsebene M2 erfolgen. Die Meta-Modellierung dient als Struktur für einen späteren Vergleich der Bussysteme.

Der Abstraktionsgrad für einen späteren Vergleich der einzelnen Bussysteme muss so gewählt werden, dass die jeweiligen bussystemabhängigen Spezifika der Modellierungsebene M1 den abstrakten Funktionalitäten der Modellierungsebene M2 eindeutig zugeordnet werden können. Hierzu ist eine geeignete Strukturierung zu wählen, auf die sich alle busspezifischen Umsetzungen der Modellierungsebene M1 abbilden lassen. Der Abstraktionsgrad kann zwischen der Abstraktion der einzelnen Funktionalitäten variieren. Er ist abhängig von den Gemeinsamkeiten der Bussysteme. Sind Funktionalitäten identisch, müssen sie kaum abstrahiert werden. Je unterschiedlicher die Funktionalitäten jedoch sind, umso abstrakter müssen sie für einen Vergleich und der nachfolgenden Vereinheitlichung abgebildet werden.

4.1.1 Lage der Schnittstelle innerhalb der Steuerungsebenen

Die Lage der zu definierenden funktional einheitlichen Applikationsschnittstelle muss später zwischen der Applikation und der Hardware liegen. Nur so kann sichergestellt werden, dass die Applikation unabhängig vom unterlagerten Bussystem ist. Die Struktur der Modellierung ist abhängig von der Sichtweise auf die Modellierungsebene M1. Es

bestehen zwei denkbare Sichtweisen, die zu einer Abstraktion der Bussysteme und einer daraus resultierenden Schnittstelle führen. Die Abstraktion kann entweder aus Sicht der Applikation oder aus Sicht der Kommunikation bzw. dem Applikationsprofil des Bussystems erfolgen (Bild 4.2).

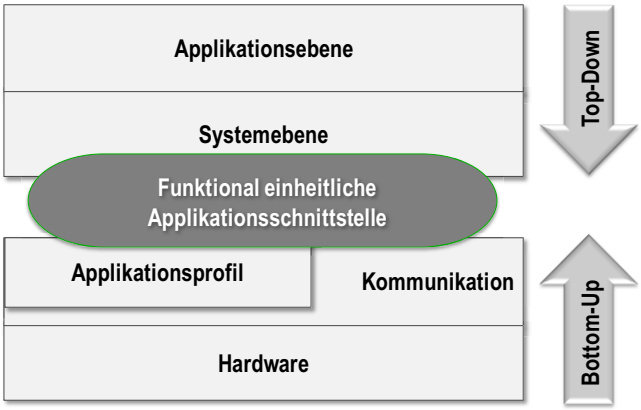


Bild 4.2: Lage der funktional einheitlichen Applikationsschnittstelle innerhalb der Steuerungsebenen

Diese beiden Sichtweisen beschreiben sich wie folgt:

- Bei der bussystemnahen Sichtweise werden die Funktionalitäten und Funktionen durch ihre Auswirkung auf das Protokoll beschrieben. Dieser Zusammenhang lässt sich für jede Funktionalität und Funktion exakt beschreiben. Diese Sichtweise kann durch einen Bottom-Up-Ansatz ausgehend von der Spezifikation erstellt werden. Dieses Extrem weist eine geringe Abstraktion im Vergleich zur Modellierungsebene M1 auf, da die Spezifikationen die Funktionalitäten sehr detailliert beschreiben. Diese detaillierten Beschreibungen führen zu sehr kleinen, wenig abstrahierten Modellen im Vergleich zur Modellierungsebene M1.
- Die anwendungsnahe Sichtweise betrachtet die Funktionalitäten und Funktionen, wie sie aus der Applikation heraus genutzt werden und sie sich auf die entsprechende Bussystemkomponente auswirkt. Hierbei kann der Top-Down-

Ansatz ausgehend von der Applikation angewendet werden. Dieses Extrem stellt eine sehr hohe Abstraktion gegenüber der Modellierungsebene M1 dar, da lediglich Funktionalitäten und insbesondere Funktionen betrachtet werden können, wie sie in der Applikation angewendet werden. Dadurch können Details, die in den Spezifikationen beschrieben sind, bei der Abstraktion verloren gehen und somit in der Meta-Modellierung fehlen.

Die in Kapitel 2 gestellten Anforderungen nach einfacher Handhabbarkeit und Bussystemunabhängigkeit sprechen für die Modellierung anhand der anwendungsnahen Sichtweise. Die Funktionalitäten werden dabei entsprechend der in Applikationen verwendeten Funktionen abgeleitet. Eine Abstraktion mit dieser Sichtweise ist allein jedoch nicht durchführbar. Zum einen gib es keine Standardapplikationen, die als Grundlagen für eine Abstraktion herangezogen werden kann. Zum anderen müssen die Anforderungen nach Universalität, Eindeutigkeit und Applikationsunabhängigkeit der vereinheitlichten Funktionalitäten und Funktionen erreicht werden.

Wird lediglich eine Modellierung anhand der bussystemnahen Sichtweise durchgeführt, werden alle Funktionen in die Schnittstelle mit aufgenommen, die für Applikationen keinerlei Relevanz haben. Dadurch wird die Schnittstelle umfangreicher als notwendig und der Grad der Vereinheitlichung sinkt. Die Handhabbarkeit ist dadurch nicht mehr gewährleistet.

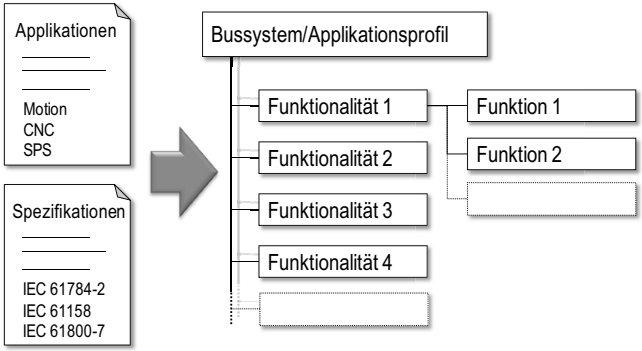


Bild 4.3: Spezifikationen und Applikationen als Basis zur Definition des strukturellen Zusammenhangs von Bussystemfunktionalitäten

Um eine geeignete und angemessene Schnittstelle zu finden muss neben der applikationsnahen auch die bussystemnahe Sichtweise betrachtet werden. Für die bussystemnahe Betrachtung werden die Spezifikationen für Ethernet-basierte Bussysteme (IEC 61784-2, IEC 61158) und Antriebsprofile (IEC 61800-7) herangezogen. Sie werden hinsichtlich ihrer Strukturen analysiert. Aus diesem Ergebnis wird eine einheitliche Struktur für Ethernet-basierte Bussysteme abgeleitet (Bild 4.3). Sie dient als Grundlage für den späteren, detaillierten Vergleich nach einer Spezialisierung. Für die applikationsnahe Betrachtung werden beispielhaft Applikationen und deren Funktionalitäten aus den Bereichen Motion, SPS und CNC herangezogen. Sie dienen dazu, die aus der bussystemnahen Sichtweise identifizierten Funktionen zu validieren.

4.2 Definition von generalisierten Bussystemfunktionalitäten und deren struktureller Zusammenhang

Betrachtet man industrielle Kommunikationsnetzwerke ausgehend von der Spezifikation Ethernet-basierter Bussysteme der IEC 61158 und der Applikation, so lassen sie sich zunächst als Sicht auf Geräteebene wie in Bild 4.4 gezeigt darstellen.

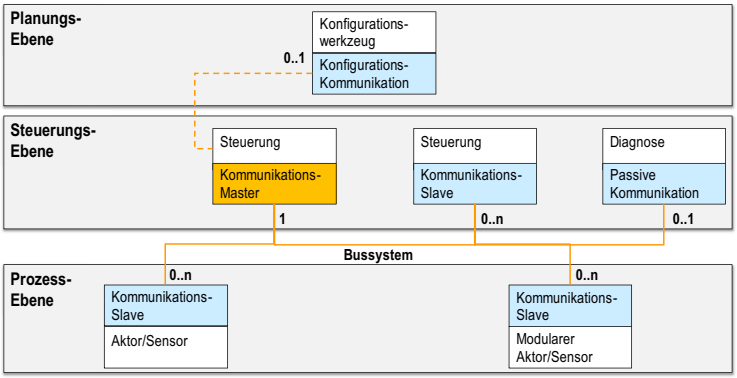


Bild 4.4: Komponenten eines industriellen Kommunikationsnetzwerks

Ein industrielles Kommunikationsnetzwerk besteht aus einem Kommunikationsmaster als zentrale Komponente. Dieser Kommunikationsmaster ist typischerweise der Steuerung zugeordnet. Die weiteren Automatisierungskomponenten sind als Kommunikati-

ons-slaves mit dem Master verbunden. Bei den Slaves kann es sich um weitere Steuerungen, geschlossene und modulare Aktoren oder Sensoren handeln. Bei geschlossenen Slaves handelt es sich um Geräte, die nicht erweiterbar sind (Beispiel Antriebsverstärker). Modulare Slaves sind Geräte, die erweitert werden können (zum Beispiel Buskoppler mit E/A-Klemmen). Es können auch passive Kommunikationsteilnehmer angeschlossen sein, die die Kommunikation nur mithören, jedoch nicht aktiv an ihr teilnehmen. Passive Kommunikationsteilnehmer werden entweder zur Diagnose der Kommunikation oder des Automatisierungsprozesses in das Netzwerk mit aufgenommen. Die Konfiguration der Kommunikation und der Applikation der jeweiligen Geräte erfolgt mit Hilfe eines Konfigurationswerkzeugs, das entweder direkt im Master integriert ist oder aber die Konfiguration an den Kommunikationsmaster übergibt.

Die Anordnung der Komponenten und deren physikalische Verbindung (Verkabelung) werden als Netzwerktopologie bezeichnet. Sie wird deshalb als übergeordnetes Strukturelement bei der Modellierung angesetzt. Als abstraktester Anwendungsfall lässt sich mit dem Top-Down-Ansatz die Kommunikation zwischen Geräten und deren Applikationen definieren. (Bild 4.5). Die Applikationen in den Geräten tauschen Daten über die Kommunikation des Ethernet-basierten Bussystems untereinander aus. Die Kommunikation ist abhängig von der Konfiguration durch den Master. Der Master stellt dabei ein besonderes Gerät dar, das für die Konfiguration der Kommunikation und der Geräte bzw. Applikationen zuständig ist. Der Master hat selbst eine oder mehrere Applikationen.

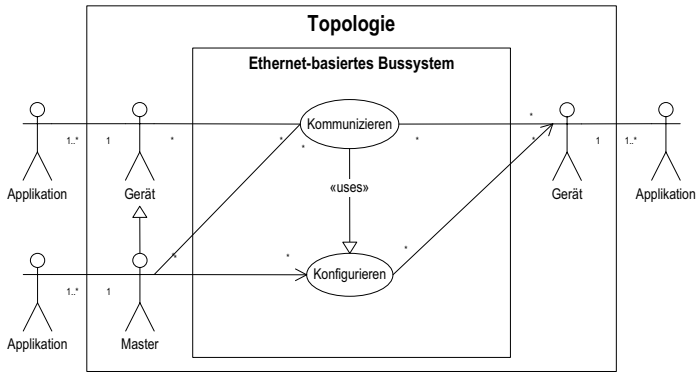


Bild 4.5: Abstrakter Anwendungsfall der industriellen Kommunikation und deren Topologie

Für eine strukturelle Beschreibung Ethernet-basierter Bussysteme lassen sich für die Kommunikation und die Geräte mit dem Bottom-Up-Ansatz weitere Funktionalitäten ableiten (Bild 4.6). Dabei findet sich die Topologie, die die Kommunikation zwischen Geräten darstellt aus dem bereits beschriebenen abstrakten Anwendungsfall wieder. Die Kommunikation lässt sich in Funktionalitäten wie Datenaustausch, Hochlauf, Hotplug, Buszugriff, Synchronisation und logische Verbindungen gliedern. Das Gerät kann in die Funktionalitäten Gerätemodell, Applikationsprofil und Master oder Slave unterteilt werden. Die Konfiguration beeinflusst dabei alle aufgeführten Funktionalitäten.

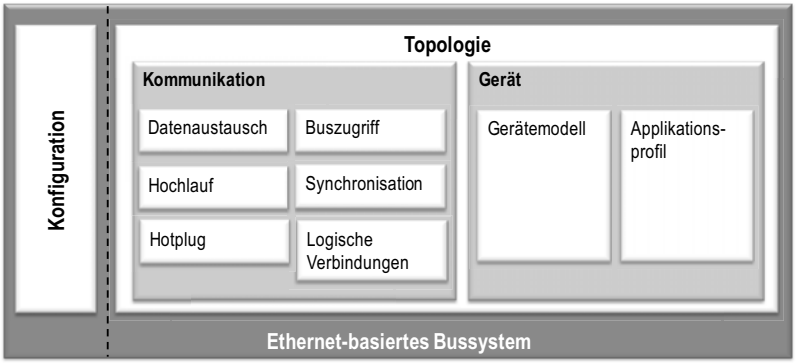


Bild 4.6: Funktionalitäten Ethernet-basierter Bussysteme

4.3 Klassifikation von Funktionalitäten und deren Struktur in einem Meta-Modell

Im Folgenden werden die einzelnen Funktionalitäten genauer beschrieben, um später anhand dieser Strukturierung den Vergleich der betrachteten Bussysteme vornehmen zu können. Die Funktionalitäten werden dazu mit Hilfe der Klassifikation aus den Spezifikationen der betrachteten Bussysteme als Meta-Modell für Ethernet-basierte Bussysteme definiert.

4.3.1 Kommunikation

Die Kommunikation ist der wesentliche Teil der Spezifikation von Ethernet-basierten Bussystemen. Hierbei unterscheiden sich die Bussysteme in ihren Mechanismen zur Er-

reichung des deterministischen Datenaustauschs. Die Struktur der Spezifikationen ist hier ebenfalls extrem unterschiedlich, da es, wie bereits im Stand der Technik erläutert, keine Zuordnung zu den Ebenen des OSI-Schichtenmodells gibt. Deshalb wird im Folgenden eine Klassifikation der Funktionen vorgenommen. Die Funktionalität der Kommunikation und deren allgemeiner struktureller Zusammenhang sind in Bild 4.7 dargestellt. Der allgemeine strukturelle Zusammenhang ist aus einer Betrachtung der einzelnen Strukturen der Bussystemspezifikationen abgeleitet.

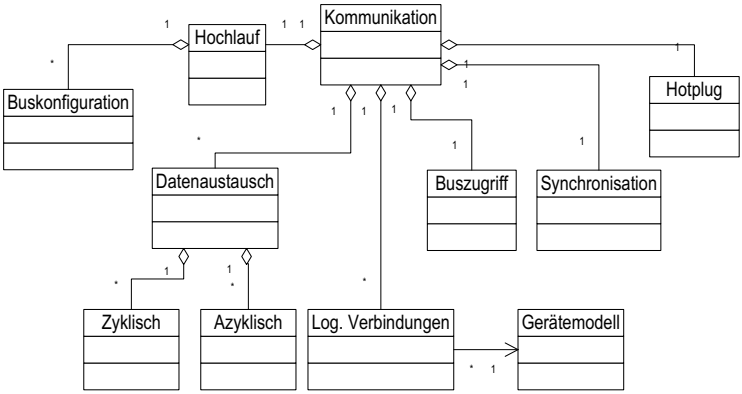


Bild 4.7: Funktionalitäten der Kommunikation und deren Struktur

Im Einzelnen lassen sich die Funktionalitäten der Kommunikation wie folgt beschreiben:

4.3.1.1 Hochlauf

Die Funktionalität des Hochlaufs stellt die Möglichkeit zur Konfiguration der Echtzeit-Kommunikation, die zwischen den Geräten stattfindet, dar. Der Hochlauf ist in mehrere Schritte unterteilt, in denen die Busteilnehmer für den deterministischen Datenaustausch konfiguriert werden. Die Konfiguration des Buszugriffs und des Telegrammaufbaus muss vor der eigentlichen Echtzeitkommunikation festgelegt werden und allen Teilnehmern durch den Kommunikationsmaster mitgeteilt werden.

Während des Hochlaufs kann bereits die Konfiguration und Parametrierung der Applikationen vorgenommen werden. Teilweise bestehen Abhängigkeiten zwischen der Kom-

munikation und der Applikation, die im späteren Vergleich der Funktionalitäten unterschiedlicher Bussysteme genauer betrachtet werden.

4.3.1.2 Logische Verbindungen

Logische Verbindungen stellen den Datenfluss zwischen den Teilnehmern aus Applikationssicht dar. Mit den logischen Verbindungen wird während des Hochlaufs festgelegt, welche Daten zwischen den einzelnen Geräten unter Echtzeitbedingungen ausgetauscht werden sollen. Hierbei existiert eine enge Verbindung zum Gerätemodell, da für die Beschreibung der logischen Verbindungen für den Echtzeitdatenaustauschs die Adressierung des Gerätemodells verwendet wird. Die logischen Verbindungen müssen vor dem Hochlauf bekannt sein, da sie Auswirkungen auf das Echtzeitverhalten der Kommunikation haben.

4.3.1.3 Datenaustausch

Der Datenaustausch eines Kommunikationssystems beschreibt, mit welchen Methoden zyklische oder azyklische Informationen zwischen Teilnehmern ausgetauscht werden können.

Bei der zyklischen Kommunikation werden die Daten in einem festen Zyklus ausgetauscht. Es ist kein spezielles Anfordern der Daten notwendig, der Datenaustausch wird fest mit Hilfe der logischen Verbindungen konfiguriert.

Beim azyklischen Datenaustausch wird definiert, wie die Anfragen zum Schreiben und Lesen von Daten erfolgen und die entsprechende Antwort generiert wird.

4.3.1.4 Buszugriff und Protokollaufbau

Die Funktionalität des Buszugriffs regelt die Zuteilung des Busses für das Senden der Daten, so dass keine Kollisionen auftreten, die das Echtzeitverhalten beeinträchtigen. Der Buszugriff wird während des Hochlaufs konfiguriert. Der Buszugriff beinhaltet auch die Festlegung des Protokollaufbaus. Jedem Teilnehmer muss bekannt sein, wie er die Daten innerhalb des Protokolls den logischen Verbindungen zuordnen und zu interpretieren hat. Es muss sichergestellt werden, dass es zu keiner Beeinflussung von Daten anderer Kommunikationsteilnehmer kommen kann.

4.3.1.5 Synchronisation

Der Buszugriff sorgt dafür, dass es zu keiner Verletzung der Echtzeitfähigkeit kommt. Die Synchronisation stellt sicher, dass alle Teilnehmer einen globalen Takt haben, an dem Sollwerte umgesetzt und Istwerte erfasst werden. Die Synchronisation stellt eine der wesentlichen Funktionalitäten für Applikationen in der Automatisierungstechnik dar. Die einzelnen Applikationen synchronisieren sich über das Bussystem aufeinander.

4.3.1.6 Hotplug

Hotplug beschreibt die Funktionalität, ein Gerät während der aktiven Echtzeitkommunikation in diese mit aufzunehmen. Das bedeutet, dass ein bisher unbekanntes Gerät an das Kommunikationssystem angeschlossen wird, das am Hochlauf der laufenden Kommunikation nicht teilgenommen hat. Dadurch fehlen diesem Gerät die Informationen zum Buszugriff, der logischen Verbindungen und dem Datenaustausch. Diese Informationen muss das Gerät vom Master nachträglich bekommen. Ebenso müssen die relevanten Konfigurationen und Parameter der Applikation übertragen werden. Dabei darf die aktive Echtzeitkommunikation nicht gestört und die anderen Kommunikationsteilnehmer nicht beeinflusst werden.

4.3.2 Gerät

Aus Applikationssicht spielt die genaue Beschreibung der Geräte und deren Verhalten eine wesentliche Rolle. Hier existiert ebenfalls keine einheitliche Beschreibung zwischen den Bussystemspezifikationen und den Applikationsprofilen. Deshalb ist auch hier zuerst eine Klassifikation der Funktionalitäten der betrachteten Systeme durchzuführen. Die Funktionalität eines Gerätes und deren Struktur, wie sie sich durch Vergleich der Strukturen in den Spezifikationen abbilden lassen, sind in Bild 4.8 dargestellt.

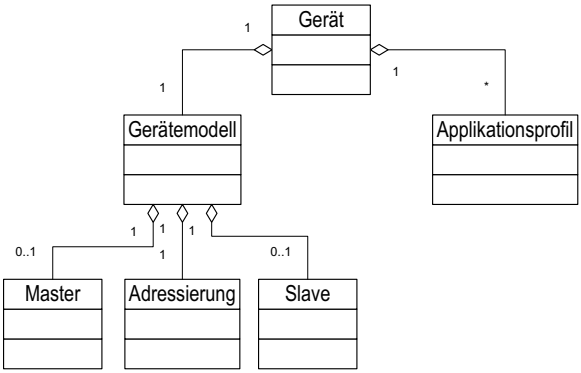


Bild 4.8: Funktionen und Struktur der Funktionalitäten der Applikation

Im Einzelnen lassen sich die Funktionalitäten des Geräts wie folgt beschreiben:

4.3.2.1 Gerätemodell

Das Gerätemodell beschreibt die logische Sicht auf ein Gerät. Es wird beschrieben, wie sich die logische Struktur des Geräts zusammensetzt und welche Komponenten adressiert werden können. Dabei kann die logische Sicht mit der baulichen Struktur des Gerätes übereinstimmen (zum Beispiel Buskoppler mit modularen E/As) oder aber es handelt sich um eine rein logische Strukturierung (zum Beispiel doppelachsiger Antriebsverstärker). Für die Adressierung werden hierarchische Beschreibungen verwendet. Es können das ganze Gerät oder einzelne Komponenten adressiert werden. Das Gerätemodell beschreibt neben der inneren Struktur auch, ob es sich um einen Master oder Slave handelt. Dies ist hauptsächlich für die Kommunikation relevant, wird aber dennoch dem Gerät zugeordnet, da es für die Topologie und die Adressierung in der Applikation relevant ist.

4.3.2.2 Applikationsprofil

Ein Applikationsprofil, wie bereits in 2.1 definiert, beschreibt Funktionalitäten aus Applikationssicht und deren Abbildung auf Funktionen bzw. Parameter (Bild 4.9). Der Aufbau eines Applikationsprofils soll an dieser Stelle genauer erläutert werden, da der Ver-

gleich und die Vereinheitlichung der Applikationsprofile für eine funktional einheitliche Applikationsschnittstelle äußerst relevant sind.

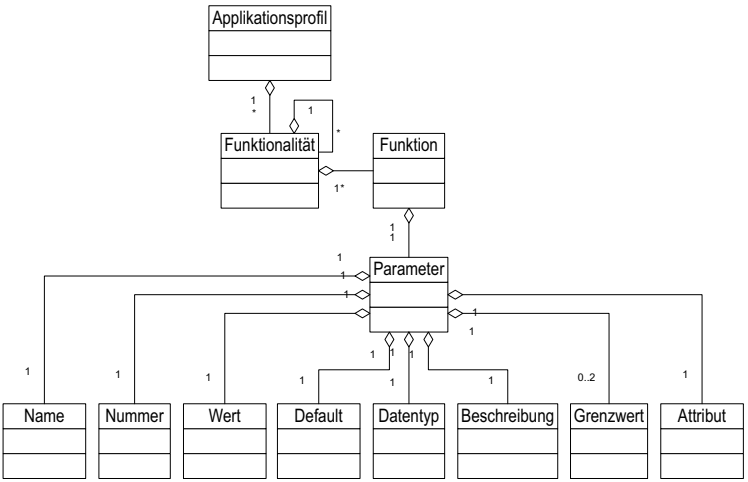


Bild 4.9: Funktionalitäten und struktureller Aufbau eines Applikationsprofils

Der strukturelle Aufbau eines Applikationsprofils gliedert sich wie die Kommunikation auch in einzelne Funktionalitäten. Diese lassen sich dann immer weiter detaillieren, bis sie sich als einzelne Funktionen, bzw. Parameter abbilden lassen. Alle betrachteten Applikationsprofile arbeiten in irgendeiner Form mit Parametern. Die Klassifikation bei der Beschreibung von Parametern stellt sich wie folgt dar:

Name
Gibt dem Parameter einen Namen, der die Funktion kurz beschreibt.

Nummer
Eine eindeutige Nummerierung, die zur Identifikation des Parameters und der Übertragung über das Bussystem dient. Die Nummerierung kann eine Strukturierung der Parameter beinhalten.

Wert
Der Wert gibt das eigentliche Datum des Parameters an. Dabei handelt sich um den ermittelten Wert durch das Gerät beim Lesen oder um den zu setzenden Wert bei einem Schreibzugriff.

Default
Gibt den Standardwert an, den ein Parameter bei der Initialisierung hat.

Datentyp
Der Datentyp legt fest, wie die binären Daten in numerische Werte interpretiert werden müssen. Unter den Datentyp fällt auch die physikalische Einheit des Parameters.

Beschreibung
Die Beschreibung ermöglicht es, den Parameter bzw. dessen Funktion über den Namen hinaus durch einen Text genauer zu beschreiben.

Grenzwert
Der Grenzwert legt die obere und untere Grenze des Wertes fest, die der Parameter annehmen kann.

Attribut
Das Attribut ermöglicht es, weitere Eigenschaften des Parameters festzulegen (zum Beispiel Einheit, Schreibschutz).

Zum späteren Vergleich der Funktionalitäten und Funktionen muss ebenfalls eine Strukturierung der Funktionen der betrachteten Applikationsprofile vorgenommen werden. Die betrachteten Applikationsprofile für digitale Antriebe lassen sich anhand der Antriebszustände und der verschiedenen Betriebsarten strukturieren. Unter Antriebszuständen sind Funktionalitäten zu sehen, die den Antrieb und dessen Verhalten allgemein beschreiben. In den Betriebsarten sind Funktionalitäten enthalten, die abhäng von der jeweiligen Reglerstruktur sind.

4.4 Zusammenfassung

In diesem Kapitel wurde erörtert, welcher Abstraktionsgrad für die Modellierung und einem nachfolgenden Vergleich von Ethernet-basierten Bussystemen und der zugehörigen

Applikationsprofile notwendig ist. Dazu wurden die zugehörigen existierenden Modelle und Beschreibungen den Modellierungsebenen zugeordnet. Als Ergebnis wird ein geeignetes Meta-Modell aus den Spezifikationen abgeleitet. Es dient als Struktur für einen nachfolgenden Vergleich.

Um dieses Meta-Modell zu definieren, sind unter Berücksichtigung der verfügbaren Beschreibungsformen (Applikationen, Applikations- und Bussystemspezifikationen) prinzipiell zwei Ansätze möglich. Der Top-Down-Ansatz stellt eine Möglichkeit dar, bei der die Abstraktion aus applikationsnaher Sicht beschrieben wird. Eine weitere Möglichkeit bietet der Bottom-Up-Ansatz an, der die Abstraktion aus bussystemnaher Sicht definiert.

Da keine vollständige Beschreibung von Funktionalitäten aus Applikationssicht existiert, werden die beiden Ansätze Top-down und Bottom-Up kombiniert. Die Spezifikationen dienen zur Sammlung der Funktionalitäten und die Sicht aus der Applikation legt die notwendige Abstraktion von oben fest.

Es werden die Funktionalitäten und deren struktureller Zusammenhang als Meta-Modell hergeleitet, anhand derer im folgenden Kapitel ein Vergleich der Ethernet-basierten Bussysteme und der Applikationsprofile durchgeführt werden. Aus dem Vergleich kann danach eine funktional einheitliche Applikationsschnittstelle abgeleitet werden.

5 Vergleich der Bussystemfunktionalitäten durch Spezialisierung

Anhand des im vorherigen Kapitel hergeleiteten Meta-Modells für Ethernet-basierte Bussysteme und Applikationsprofile wird im Folgenden durch Spezialisierung der einzelnen Funktionalitäten ein Vergleich durchgeführt. Dazu werden die Funktionalitäten jedes Systems durch Spezialisierung detailliert beschrieben und gegenübergestellt. Aus dieser Gegenüberstellung lassen sich im darauffolgenden Kapitel die Funktionalitäten und Funktionen für eine einheitliche Applikationsschnittstelle ableiten.

5.1 Topologie

Zunächst wird, wie bereits im vorherigen Kapitel, die Topologie betrachtet. Hierbei wird zwischen der physikalischen und der logischen Topologie unterschieden. Die physikalische Topologie beschreibt, wie die Ethernet-Kabel zwischen den einzelnen Geräten innerhalb eines Netzwerkes verlaufen können. Die logische Topologie beschreibt den Telegrammfluss innerhalb des Netzwerkes. Sie zeigt, wie die Telegramme zwischen den jeweiligen Geräten übertragen werden.

Bei der physikalischen Topologie ist bei den betrachteten Systemen die Verkabelung nach Bild 5.1 möglich. Ethercat und Profinet erlauben eine Netztopologie nach einer Verdrahtung als Linie, Stern, Baum und Linie. Profinet bietet zusätzlich die Möglichkeit, speziell zu konfigurierende Switches einzusetzen. Sercos lässt sich als Linie oder Ring verdrahten. Alle der betrachten Bussysteme können redundante Kommunikationspfade zwischen den Teilnehmern und dem Master aufbauen. Bei Ethercat und Sercos wird dies bei der Topologie Ring erreicht. Ethercat hat spezielle Anforderungen an den ersten Slave nach dem Master. Dieser muss zur Berechnung der Zeitschlitze besonders konfiguriert werden. Bei Profinet muss ein spezieller Redundanzmanager in das Netzwerk eingebracht werden.

Die logische Topologie entspricht bei Profinet exakt der physikalischen, da genau jeder Teilnehmer ein Telegramm auf einer bestimmten Route erhält. Bei Sercos und Ethercat wird die logische Topologie immer zu einem Ring. Bei diesen beiden Bussystemen werden die Telegramme von einem zum nächsten Gerät weitergeleitet und zum Master zurück gesendet. Bei einem Doppelring werden die Telegramme in zwei Richtungen übertragen. Für eine logische Topologie als Doppelring muss physikalisch ein Ring vorhan-

den sein. Eine genauere Betrachtung des Telegrammaufbaus und Buszugriffs erfolgt im Kapitel 5.2.2.

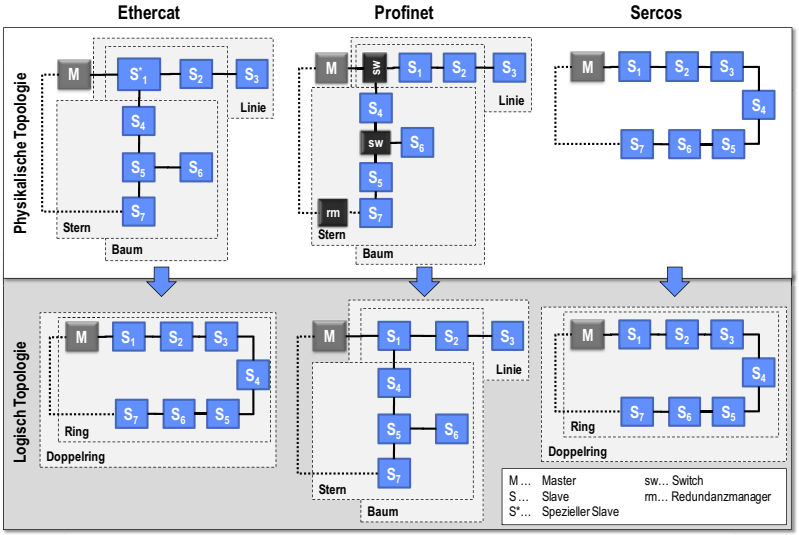


Bild 5.1: Topologie der betrachteten Ethernet-basierten Bussysteme

Durch den Einsatz von Switches lässt sich mit Profinet nahezu jede beliebige Topologie umsetzen. Allerdings ist dadurch ein hoher Konfigurationsaufwand der Switches notwendig und eine Redundanz nur durch einen speziellen Redundanzmanager zu erreichen.

5.2 Kommunikation

5.2.1 Hochlauf

Die einzelnen Hochläufe der betrachten Ethernet-basierten Bussysteme lassen sich als Zustandsdiagramme darstellen (Bild 5.2). Der Hochlauf ist in einzelne Zustände unterteilt, in denen verschiedene Möglichkeiten zur Kommunikation vorhanden sind.

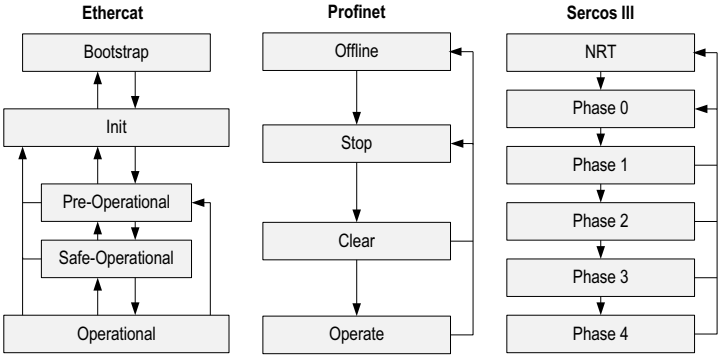


Bild 5.2: Zustandsdiagramme der Hochläufe zur Konfiguration der Kommunikation

Ethercat definiert fünf Zustände für den Hochlauf. Diese werden im Normalfall ausgehend von *Init* bis nach *Operational* sequenziell durchlaufen. Der Zustand *Bootstrap* ist optional und muss nicht vorhanden sein. Erfordert es die Applikation oder treten Fehler auf, sind Rücksprünge zu den vorhergehenden Zuständen möglich.

Profinet beschreibt vier Zustände, die ebenfalls bei einem normalen Hochlauf von *Offline* nach *Operate* sequentiell durchlaufen werden. Rücksprünge zu vorhergehenden Zuständen sind nur über den Zustand *Offline* möglich.

Sercos III beschreibt sechs Phasen, über die der Hochlauf von *NRT* (Non Real Time) bis zur *Phase 4* erfolgt. Sind Rücksprünge notwendig, dann sind diese nur über die Zustände *NRT* und *Phase 0* möglich.

Die einzelnen Zustände des Hochlaufs unterscheiden sich hinsichtlich ihrer Kommunikationsmöglichkeiten. Direkt nach dem Einschalten eines Geräts ist keine Möglichkeit zur Kommunikation vorhanden.

Die erste Kommunikationsmöglichkeit nach dem Einschalten stellt die Standard-Ethernet-Kommunikation dar. Dabei werden Protokolle der OSI-Schicht 3 übertragen. Dies wird beispielsweise zur Übertragung von Firmwaredateien an Geräte oder den Zugriff auf Webserver verwendet.

Um eine eindeutige Zuordnung zu den Geräten zu haben, ist eine Adressvergabe notwendig. Bei der Adressvergabe können auch fest eingestellte Adressen in den Geräten durch den Master abgefragt werden.

Die azyklische Kommunikation dient zur Konfiguration der zyklischen Kommunikation und der Synchronisation. Zusätzlich können über die azyklische Kommunikation Applikationsparameter übertragen und somit die Applikation konfiguriert und parametriert werden. Ebenso ist es möglich, Applikationsparameter auszulesen, die nicht echtzeitkritisch sind.

Bei der zyklischen, synchronen Kommunikation werden die Daten zu fest vorgegebenen Zeiten zwischen den Teilnehmern übertragen. Werden die Sollwerte als gültig gesetzt, dann werden die digitalen Daten von den Aktoren in physikalische Größen umgesetzt.

Eine Zuordnung der Kommunikationsmöglichkeiten zu den einzelnen Zuständen der betrachteten Bussysteme ist in Tabelle 5.1 dargestellt. Dabei wird ersichtlich, dass keine grundlegenden Unterschiede zwischen den einzelnen Systemen bestehen. Lediglich Sercos III hat einen separaten Zustand für die Adressvergabe, während dieser bei Ethercat keine Standard-Kommunikation möglich ist. Bei Profinet gibt es keinen eigenen Zustand in dem keine azyklische Kommunikation möglich ist. In den möglichen Zustandsübergängen unterscheiden sich Sercos III und Profinet gegenüber Ethercat grundlegend.

	Ethercat					Profinet				Sercos III					
	Bootstrap	Init	Pre-Operational	Safe-Operational	Operational	Offline	Stop	Clear	Operate	NRT	Phase 0	Phase 1	Phase 2	Phase 3	Phase 4
Zustand nach dem Einschalten	(X)	X				X				X					
Standard-Ethernet Kommunikation	X		X	X	X	X	X	X	X	X	X	X	X	X	X
Adressvergabe		X					X					X			
Azyklische Kommunikation			X	X	X		X	X	X				X	X	X
Zyklische, synchrone Kommunikation				X	X			X	X					X	X
Sollwerte gültig					X				X						X

Tabelle 5.1: Zusammenfassung der Zustände der jeweiligen Hochläufe

5.2.2 Buszugriff und Protokollaufbau

Der Buszugriff bei Ethernet-basierten echtzeitfähigen Bussystemen erfolgt nicht durch das sonst bei Ethernet verwendete Verfahren CSMA/CD. Alle betrachteten Systeme haben spezielle Erweiterungen auf Ebene zwei des OSI-Schichtenmodells, da die Prioritätsvergabe von Ethernet für einen jitterfreien Determinismus nicht ausreichend ist. Der Master steuert den Buszugriff für alle Teilnehmer. Für einen Vergleich werden die Buszugriffsverfahren der einzelnen Busse anhand eines Kommunikationszyklus genauer betrachtet. Die einzelnen Buszugriffsverfahren sind in Bild 5.3 dargestellt. Ein Kommunikationszyklus stellt eine Periode der zyklischen Kommunikation dar.

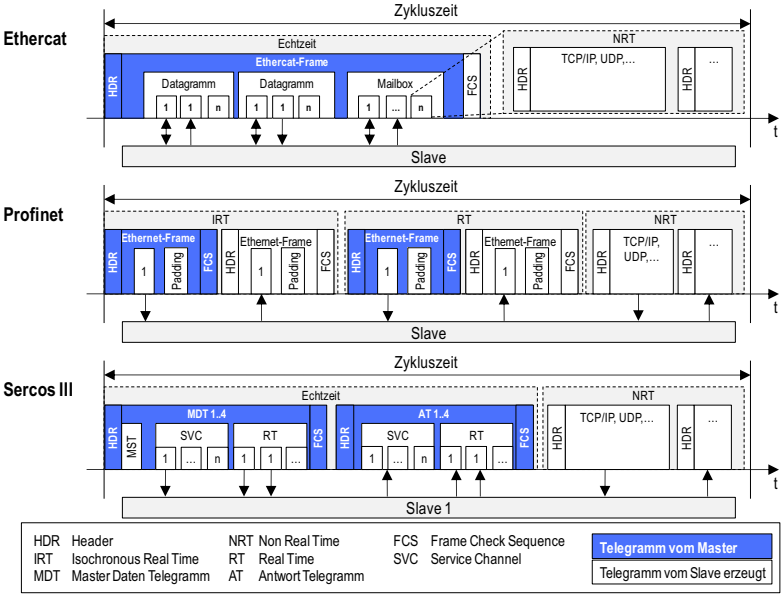


Bild 5.3: Übersicht der Buszugriffsverfahren und des Protokollaufbaus

Bei Ethercat erzeugt der Master ein Ethernet-Telegramm zu Beginn des Kommunikationszyklus. Dieses Ethernet-Telegramm enthält die Broadcast-MAC-Adresse, so dass er von allen Teilnehmern ausgewertet wird. Es sind Daten für mehrere Teilnehmer innerhalb dieses Telegramms enthalten und wird deshalb auch als Sammeltelegramm be-

zeichnet. Die Anzahl der generierten Telegramme hängt von den zu übertragenen Daten innerhalb des Kommunikationszyklus ab. Übersteigt die Summe aus Nutzdaten und Rahmen des Ethernet-Telegramms 1518 Bytes (max. Telegrammlänge für Ethernet), so wird ein zweites Telegramm generiert.

In diesen Ethernet-Telegrammen sind ein Ethercat-Header und mehrere Datagramme enthalten. Der Ethercat-Header beschreibt den Typ und die Länge der nachfolgenden Datagramme. Jedes dieser Datagramme enthält Daten für die einzelnen Teilnehmer. Wo sich diese innerhalb des Telegramms befinden, wurde während des Hochlaufs mitgeteilt. Der Slave reicht das Telegramm an seinen Ports weiter. Dabei entnimmt er die zyklischen Sollwerte und trägt seine Istwerte ein. Ob die Daten vom Master oder einem anderen Slave stammen, spielt keine Rolle. Somit lässt sich auch die Kommunikation zwischen einzelnen Slaves ermöglichen. Das Eintragen der Daten durch den Slave kann an derselben Stelle im Telegramm sein, an der sich die Daten für ihn vom Master befinden. Für den azyklischen Datenaustausch nutzt Ethercat Mailboxen. Diese werden ebenfalls vom Teilnehmer ausgewertet. Dies muss jedoch nicht in jedem Zyklus erfolgen. Am Ende des durchlaufenden Telegramms berechnet der Slave die Prüfsumme neu. Der Austausch von Standard-Ethernet-Telegrammen bei Ethercat erfolgt durch Tunneln über die Mailboxen. Das bedeutet, die Standard-Ethernet-Telegramme werden in der Mailbox des jeweiligen Teilnehmers übertragen, der sie erhalten soll.

Bei Profinet erzeugt der Master für jeden Teilnehmer ein separates Telegramm. Dafür stehen innerhalb eines Kommunikationszyklus drei Zeitabschnitte zur Verfügung. Innerhalb des ersten Zeitabschnitts werden die Daten mit höchster Echtzeitanforderung (IRT Isochronous Real Time) übertragen. Dabei hat jeder Teilnehmer einen festen Zeitschlitz, in dem er das Telegramm mit den Sollwerten empfängt und in einem neuen Telegramm die Istwerte überträgt. Diese Zeitschlitze werden ebenfalls während der Konfiguration an die Teilnehmer übergeben. In einem zweiten Abschnitt werden die Echtzeitdaten (RT Real Time) mit geringeren deterministischen Anforderungen übertragen. Hierbei handelt es sich um Telegramme, die die Prioritäten von Standard-Ethernet nutzen. Innerhalb des RT-Abschnitts werden auch die azyklischen Anfragen an die Slaves gestellt. Zusätzlich können innerhalb dieses Abschnitts die Slaves durch Ereignisse (Events) Zustandsänderungen mitteilen. Im NRT-Abschnitt ist das Senden von Standard-Telegrammen ebenfalls möglich. Sie werden durch eine niederere Priorität gegenüber den Echtzeitdaten gekennzeichnet.

Bei Sercos III werden ähnlich wie bei Ethercat Telegramme durch den Master erzeugt. Hierbei sind die Daten jedoch zwischen zwei Telegrammtypen, einem MDT (Master-Daten-Telegramm) und einem AT (Antwort-Telegramm) aufgeteilt. In das MDT trägt der Master seine Daten ein. Über das AT schicken die Slaves ihr Istwerte als Antwort zum Master oder die Daten zur Kommunikation zwischen den Slaves. Die Anzahl der Telegramme hängt ebenfalls vom Umfang der Datenmenge und der damit verbundenen Telegrammlänge ab. Innerhalb dieser beiden Telegrammtypen hat jeder Slave ein SVC-Feld (Service-Channel), über das die azyklische Kommunikation stattfindet. Bei Sercos III werdend die Telegramme ebenfalls währen des Empfanges sofort weitergeleitet.

Die jeweiligen Buszugriffsverfahren und der Protokollaufbau unterscheiden sich recht stark in ihrer Umsetzung. In Tabelle 5.2 sind die wesentlichen Unterschiede zusammengefasst.

	Ethercat	Profinet IRT	Sercos III
Telegrammerzeugung nur durch Master	X	-	X
Aktives Senden der Slaves in Zeitschlitzen	-	X	-
Sammeltelegramme	X	-	X
Aktives Senden von Ereignissen der Slaves	-	X	-
Paralleles Übertragen von Standard-Ethernet	X	X	X
Erhöhung der Buslast bei zusätzlichem Teilnehmer mit Soll- und Istwerten	max {Soll-, Istwerte} + Mailbox	2 x min. Ethernet-Framelänge (64Byte)	Sollwerte + Istwerte + SVC

Tabelle 5.2: Übersicht der Buszugriffverfahren und des Telegrammaufbaus

Bei Ethercat und Sercos III gibt jeweils der Master die Telegramme vor und die Slaves tragen lediglich ihre Daten an der vorgegebenen Stelle im Telegramm ein. Statusänderungen können sie nur auf Nachfrage oder durch zyklisch konfigurierte Statuswörter mitteilen. Bei Profinet antworten die Slaves aktiv in festen Zeitrastern und können selbst Ereignisse übertragen. Aufgrund der Sammeltelegramme bei Ethercat und Sercos III wird die verfügbare Bandbreite effizienter genutzt. Ein weiterer Teilnehmer im Netzwerk bedeutet lediglich eine Erhöhung der Daten um die Nutzdaten sowie die Mailbox,

bzw. dem SVC und wenige Status- und Steuerworte. Wird dabei die maximale Telegrammlänge von 1518 Bytes überschritten, kommt es zu einer nichtlinearen Erhöhung der Daten durch zusätzliche Ethernet-Telegramme. Bei Profinet sind pro Teilnehmer generell mindestens zwei zusätzliche Ethernet-Telegramme mit je mindestens 64 Bytes (Minimale Telegrammlänge bei Ethernet) pro Kommunikationszyklus erforderlich.

5.2.3 Synchronisation

Neben dem Buszugriff und der Übertragung der Daten innerhalb eines Kommunikationszyklus, müssen die Teilnehmer synchronisiert werden. Die Telegramme erreichen zu unterschiedlichen Zeitpunkten die Teilnehmer im Netzwerk. Die Umsetzung der Sollwerte und das Erfassen der Istwerte sollen jedoch gleichzeitig zu einem globalen Zeitpunkt erfolgen. Dazu ist eine Synchronisation der Busteilnehmer im Netzwerk erforderlich.

Bei Sercos III synchronisieren sich die Busteilnehmer auf das erste gesendete MDT im Zyklus. In diesem ersten Telegramm liegt das MST (Master Synchronisations-Telegramm), auf das alle zeitlichen Offsets im Zyklus bezogen sind (Bild 5.4). Beim MST handelt es sich trotz der Bezeichnung nicht um einen Ethernet-Telegramm, sondern um 4 Bits im ersten MDT. Die Laufzeiten zwischen den Slaves bestimmt der Master in Phase 2 des Hochlaufs und berechnet daraus die Offsets und teilt sie jedem einzelnen Slave mit.

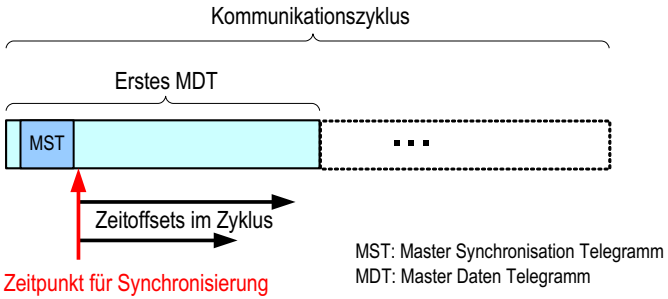


Bild 5.4: Synchronisationszeitpunkt im Sercos-Zyklus

Profinet und Ethercat hingegen nutzen keine relativen Zeiten abhängig vom Eintreffen eines speziellen Telegramms. Sie nutzen das Synchronisationsverfahren für verteilte Uhrzeiten PTP (Precision Time Protocol) nach IEC/IEEE 61588 /91/. Jeder Slave verfügt über eine Uhr, die durch regelmäßige Laufzeitmessungen und Austausch der Uhrzeiten synchronisiert werden (Bild 5.5). Die Masteruhr, auf die sich die anderen Teilnehmer synchronisieren, muss nicht allein im Master enthalten sein. Dadurch besteht keine Anforderung an das zeitlich exakte Senden der Telegramme vom Master aus. Die Daten müssen lediglich rechtzeitig beim Slave ankommen.

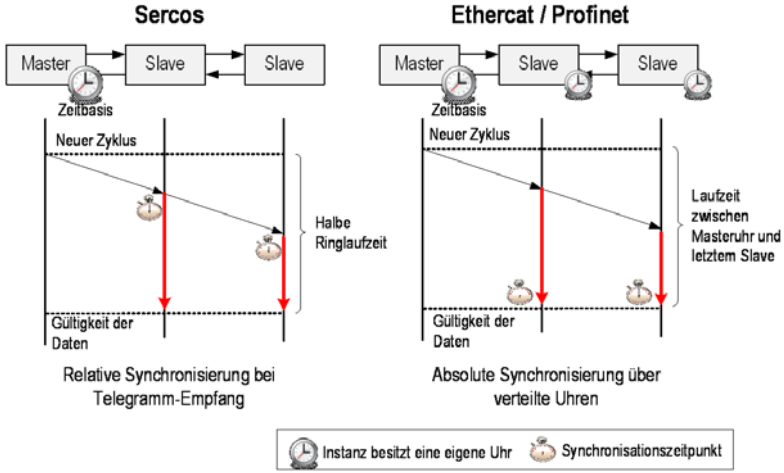


Bild 5.5: Synchronisation durch Telegramm-Empfang und verteilte Uhrzeiten

5.2.4 Datenaustausch

Die Funktionalität des zyklischen Datenaustausches lässt sich bei allen betrachteten Systemen als Funktion des Producer/Consumer-Modells (Bild 5.6) beschreiben. Der Master legt bei der Konfiguration die Beziehungen in Form von logischen Verbindungen zwischen den Geräten fest. Hierbei können produzierte Daten auch von mehreren Teilnehmern konsumiert werden. Der Datenaustausch der jeweiligen logischen Verbindung erfolgt in fest vorgegebenen Zyklen, die ein Vielfaches der Buszykluszeit sein müssen.

Bild 5.6: Producer/Consumer-Modell für den zyklischen Datenaustausch

Funktionalitäten des azyklischen Datenaustauschs lassen sich mit den Funktionen des Modells zur Client/Server-Interaktion (Bild 5.7) beschreiben. Diese Modellierung passt zu allen der betrachteten Systeme, wie sie in den Spezifikationen dargestellt sind. Dabei ist der Server im Slave integriert, der auf die Anfragen des Clients im Master antwortet.

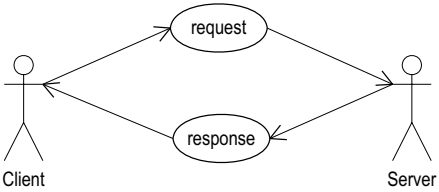


Bild 5.7: Client/Server Interaktion für den azyklischen Datenaustausch

Die Besonderheit der Funktion Event (Bild 5.8) bei Profinet, bei der Slavegeräte aktiv antworten, wird durch das aktive Senden eines Telegramms durch den Server im Slave realisiert.

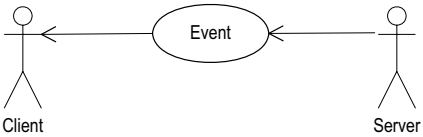


Bild 5.8: Event-Meldung als aktives Senden des Servers im Slave

5.2.5 Logische Verbindungen

Ethercat verwendet für den Datenaustausch einen virtuellen Adressraum. Hierbei handelt es sich um einen bis zu 4 Gigabyte großen Speicher. Dieser wird als eine Art Schieberegister über den Bus übertragen. Den Slaves wird mitgeteilt, an welcher Adresse sie innerhalb dieses Speichers ihre Daten vorfinden, bzw. eintragen müssen. Für die einzelnen Speicherbereiche lassen sich die Zykluszeiten als Vielfaches der Buszykluszeit festlegen mit denen der Austausch der Daten erfolgt. Dadurch können auch logische zyklische Verbindungen zwischen Slavegeräten realisiert werden. Diese sind jedoch immer nur in eine Richtung, in der das Telegramm verläuft, möglich. Für die Kommunikation zu einem in Telegrammrichtung vorhergehenden Slave ist dann ein Umkopieren im Master für den nächsten Kommunikationszyklus notwendig.

Bei Sercos III werden die logischen Verbindungen durch die Zuweisung von Parametern zwischen zwei Geräten beschrieben. In einer Liste des Gerätes ist beschrieben, welche Parameter als logische zyklische Verbindung konfiguriert werden können. Die logische Verbindung beinhaltet Informationen, in welcher Zykluszeit die Daten ausgetauscht werden sollen. Diese muss ein Vielfaches der Buszykluszeit sein. Es lassen sich somit ebenfalls Verbindungen zwischen Slavegeräten umsetzen. Durch die logische Topologie als Doppelring ist kein Umkopieren im Master notwendig.

Profinet beschreibt die Daten, die als logische Verbindungen zwischen Geräten aufgebaut werden können, als Signale. Diese Signale werden entweder auf definierte Standardtelegramme abgebildet oder aber in ein frei konfigurierbares Telegramm eingebunden. Der Telegrammaufbau wird während der Konfiguration den Slaves mitgeteilt. Die Struktur für die logische Verbindung vom Master zum Slave kann, wie bei den anderen Systemen auch, unterschiedlich zur Verbindung vom Slave zum Master sein. Handelt es sich bei Profinet bei einer logischen Verbindung um IRT, so müssen die beteiligten Switches ebenfalls während des Hochlaufs konfiguriert werden.

Trotz der Unterschiede bei den logischen Verbindungen ist eine Vereinheitlichung hin zur Applikation sowohl bei der Konfiguration, als auch beim Datenaustausch selbst möglich.

5.2.6 Hotplug

Alle betrachteten Systeme unterstützen die Möglichkeit, Geräte hinzuzufügen, während die Echtzeitkommunikation aktiv ist. Dies wird als Hotplug bezeichnet. Bei Ethercat und Sercos III muss hierzu eine bestehende physikalische und damit auch die logische Verbindung aufgetrennt werden (Bild 5.9).

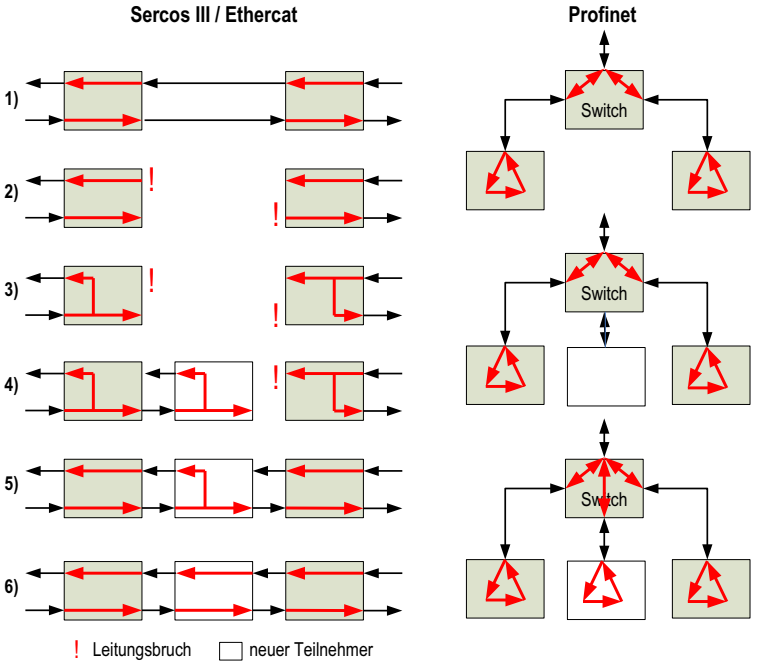


Bild 5.9: Gegenüberstellung der Methoden zur Integration von neuen Teilnehmern in die Kommunikation (Hotplug)

Voraussetzung dafür ist eine Redundanz, die einen zweiten Weg vom Master zu den abgetrennten Slaves ermöglicht. An der Stelle, an der die Trennung erfolgt, werden die Telegramme mit Hilfe der Verkabelung nach der Fullduplex-Methode /92/ auf demselben Kabel zurückgesendet (logischer Ring). Der Slave teilt den Bruch dem Master mit und meldet sich, wenn er auf dem freien Port wieder Telegramme mit dem Ethertyp des Bus-

systems empfängt. Der Master kann dann den Slave an der Trennstelle und den neuen Slave so ansteuern, dass die ursprüngliche, bzw. neue logische Verbindung hergestellt werden kann. Der neue Teilnehmer muss dem Master jedoch vorher bekannt gewesen sein und die für ihn benötigten Echtzeitdaten müssen im Telegramm vorgesehen sein.

Bei Profinet können neue Teilnehmer entweder an das Ende einer Linie oder einen Switch angeschlossen werden. Hierbei ist es jedoch nur möglich, Busteilnehmer anzubinden, die keine IRT-Daten benötigen. Die Einbindung von Busteilnehmern mit IRT-Daten erfordert einen erneuten Hochlauf der Kommunikation zur Konfiguration der Switches.

Allen betrachteten Bussystemen ist gemein, dass es nur möglich ist einen vorher bekannten Teilnehmer mit in die Echtzeitkommunikation mit aufzunehmen.

5.3 Applikationsprofile

5.3.1 Parametermodell

Der Datenaustausch und Aufruf von Funktionen zwischen den jeweiligen Applikationen erfolgt durch Parameter. Die Zusammensetzung der jeweiligen Parametermodelle wird im Folgenden zwischen den einzelnen Applikationsprofilen verglichen. Dabei werden zunächst die allgemeinen Strukturen betrachtet. Im Folgenden werden die Antriebsprofile analysiert und verglichen.

Name

Alle betrachteten Applikationsprofile ermöglichen es, die Parameternamen über das Bussystem als String auszulesen. Es existieren lediglich Unterschiede in der maximal möglichen Länge der Namen (Tabelle 5.3).

	CiA 402	CIPMotion	Profidrive	FSP Drive
Länge	beliebig	beliebig	16 Byte	60 Byte
Datentyp	String	String	String	String

Tabelle 5.3: Parameternamen der betrachteten Applikationsprofile

Nummer

Für die Adressierung von einzelnen Parametern für den zyklischen und azyklischen Datenaustausch über die Applikationsprofile wird jedem Parameter eine bestimmte Nummer zugeordnet (Bild 5.10).

Die Nummerierung der Parameter erfolgt beim Antriebsprofil CIA 402 über den Index. Über den Sub-Index können weitere Parameter unterhalb eines Index angesprochen werden. Der Sub-Index dient zur Strukturierung der Parameter. Herstellerspezifische Parameter haben einen reservierten Indexbereich, in dem keine spezifizierten Parameter existieren.

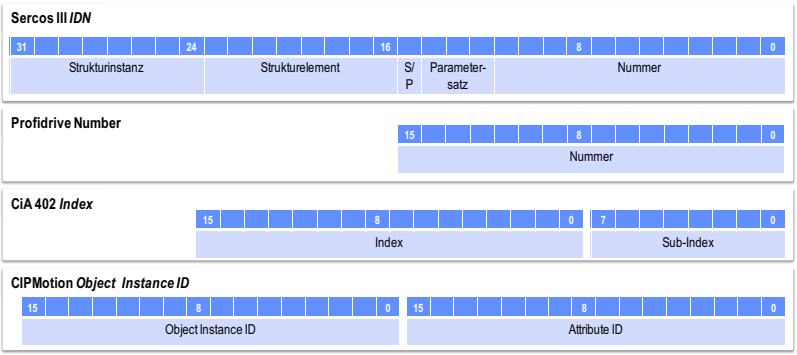


Bild 5.10: Nummerierung von Parametern der Applikationsprofile

CIP Motion bietet eine ähnliche Strukturierung, bei der Objekte adressiert werden. Diese Objekte können ebenfalls mehrere Attribute haben. Für herstellerspezifische Objekte sind spezielle Adressbereiche reserviert.

Profidrive beschreibt eine rein flache Struktur aus Parameternummern. Eine weitere Strukturierung ist nicht vorgesehen. Herstellerspezifische Parameter haben reservierte Bereiche bestimmter Parameternummern.

Bei Sercos gliedert sich die Nummerierung in die IDN (Identification Number) und untergliederte Strukturelemente. Bei jedem Parameter wird zwischen genormten S-

Parametern (Sercos-Parameter) und herstellerspezifischen P-Parametern (Producer-Parameter) unterschieden. Die Parameter lassen sich jeweils in bestimmte Parametersätze abbilden. Das bedeutet, ein Parameter kann bis zu acht verschiede Werte haben, die dann durch globales Umschalten der Parametersätze gewechselt werden. Zusätzlich bietet Sercos Strukturinstanzen. Dabei handelt es sich allerdings um eine Adressierung für modulare Geräte. Sie wird deshalb später bei der Betrachtung der Gerätemodelle weiter berücksichtigt.

Wert

Jeder Parameter hat einen bestimmt Wert. Dieser Wert kann von anderen Kommunikationsteilnehmern abhängig von den Zugriffsrechten gelesen oder geschrieben werden. Der Wert ist unabhängig davon, ob er azyklisch oder zyklisch übertragen wird.

Alle betrachten Applikationsprofile können unter einer Parameternummer einen einzelnen oder auch mehre Werte als Array hinterlegen. Der Wert ist entsprechend des Datentyps zu interpretieren.

Default

Die Parameter innerhalb eines Gerätes besitzen Standardwerte, die bei Auslieferung eingestellt sind. Bei dem Antriebsprofil CiA 402 kann dieser Standardwert zu jedem Parameter vom Gerät ausgelesen werden und somit gegebenenfalls zurückgeschrieben werden. Profidrive bietet die Möglichkeit, einen Parameter auf den Standardwert zurückzusetzen, ohne ihn jedoch auslesen zu können. CIP Motion und Sercos FSP Drive bieten standardmäßig keine dieser Funktionen.

Datentyp

Die informationstechnische Interpretation des übertragenen Werts ist durch den Datentyp festgelegt. Dieser gibt an, wie aus dem Datenfeld im Protokoll ein numerischer Wert zu bilden ist. Hierbei beschreiben die einzelnen Applikationsprofile die in Tabelle 5.4 dargestellten Datentypen. Sie lassen sich in Standarddatentypen, Text, Zeiten und weitere Datentypen einteilen.

	CiA 402	CIP Motion	Profidrive	Sercos FSP Drive
Standard-datentypen	- Unsigned8 - Unsigned16 - Unsigned32 - Unsigned64 - Integer8 - Integer16 - Integer32	- Boolean - Bitstring8 - Bitstring16 - Bitstring32 - Bistring64 - Unsigned8 - Unsigned16 - Unsigned32 - Unsigned64 - Integer8 - Integer16 - Integer32 - Integer64 - Float32 - Float64	- Boolean - BitString16 - Unsigned8 - Unsigned16 - Unsigned32 - Unsigned64 - Integer8 - Integer16 - Integer32 - Integer64 - Float32 - Float64	- Bitstring16 - Bitstring32 - Bistring64 - Unsigned16 - Unsigned32 - Unsigned64 - Integer16 - Integer32 - Integer64 - Float32 - Float64
Text	- String (IEC 646)	- String (IEC 646)	- String (IEC 646)	- String (IEC 646)
Zeiten	-	-	- Date - TimeofDay - TimeDifference - TimeConstants16 - TimeConstants32	-
Weitere	-	-	- NormalisedValue16 - NormalisedValue32 - Nibble32 - FixedPoint16 - FixedPoint32	-

Tabelle 5.4: Datentypen der Parameter unterschiedlicher Applikationsprofile

Bei den Standarddatentypen besteht eine recht große Übereinstimmung zwischen den betrachteten Applikationsprofilen. Sie dienen zum Austausch des Wertes eines Parameters.

Bei dem Austausch von Texten arbeiten alle betrachteten Systeme mit dem Datentyp String.

Bei der Definition von Datentypen für die Darstellung von Zeiten beschreibt Profidrive eine Vielzahl unterschiedlicher Datentypen. Bei den anderen Applikationsprofilen werden sie durch die Standarddatentypen und den entsprechenden Einheiten der Parameter abgedeckt.

Bei Datentypen für die binäre Darstellung von Parametern sind bei Profinet spezielle Datentypen definiert, die die anderen Applikationsprofile durch Standarddatentypen abdecken. Sie ergänzen die Parameterbeschreibung um eine Erläuterung, wie die einzelnen Bits zu interpretieren sind.

Die Interpretation der Daten als physikalische Einheit wird in den betrachteten Applikationsprofilen für jeden einzelnen Parameter beschrieben. Zu der physikalischen Einheit ist festgelegt, welchen Datentyp der jeweilige Parameter hat. Diese Information ist auch auf dem Gerät verfügbar, um die Daten richtig darzustellen und bei der Übertragung die Länge im Telegramm vorzusehen.

Beschreibung

Profidrive bietet als einziges Profil die Möglichkeit, eine Beschreibung eines Parameters aus dem Gerät auszulesen. Diese Beschreibung ist zusätzlich zum Namen des Parameters vorhanden und kann bis zu 64 Byte, bzw. Zeichen lang sein.

Grenzwert

Der Wert eines Parameters ist teilweise nur in bestimmten Wertebereichen sinnvoll oder darf gewisse Grenzen nicht überschreiten. Hierzu definieren die Profile FSP Drive, Profidrive und CiA 402 für jeden Parameter eine untere und obere Grenze für den möglichen Wertebereich. Der Wertebereich des Datentyps kann somit eingeschränkt werden. Bei CIP Motion können diese Grenzen nicht separat abgefragt werden. Es ist prinzipiell möglich, den gesamten Wertebereich des Datentyps zu schreiben. Ungültige Werte werden erst beim Schreibversuch mit einem Fehler abgelehnt.

Attribut

Die einzelnen Parameter besitzen Attribute. Diese Attribute geben an, welche azyklischen Zugriffsmöglichkeiten auf einen Parameter existieren. Hierbei wird bei Sercos unterschieden, in welchen Zuständen der Kommunikation der Parameter schreibgeschützt ist. Bei Profinet wird lediglich die aktuelle Zugriffsmöglichkeit durch ein einzelnes Bit kenntlich gemacht. Bei CIP Motion sind die Zugriffsmöglichkeiten für jeden Parameter spezifiziert und können nicht online vom Gerät ausgelesen werden. Der Zustand ist für alle Kommunikationszustände gleich. Bei CiA 402 gibt es zusätzlich das Attribut Konstante. Eine Konstante kann weder vom Gerät selbst noch über die Kommunikation geändert werden.

Verfügbare Parameter

Alle der betrachteten Applikationsprofile bieten eine Funktion an, um eine Liste aller verfügbaren Parameter aus dem Gerät auszulesen.

5.3.2 Gerätemodell

Für die Adressierung und die logische Beschreibung eines Geräts spezifizieren die Applikationsprofile unterschiedliche Gerätemodelle.

Das Sercos -Gerätemodell beschreibt die Sichtweise auf ein Gerät wie in Bild 5.11 dargestellt. Das Gerät (Device) hat eine Kommunikationsschnittstelle als Sercos III Interface. Dieses Gerät kann mehrere Slaves haben, die einem Sub-device entsprechen. Ein Antrieb entspricht bei Sercos genau einem Slave. Ein Sub-device kann wiederum aus mehreren Ressourcen bestehen, die über Instanzen adressiert werden.

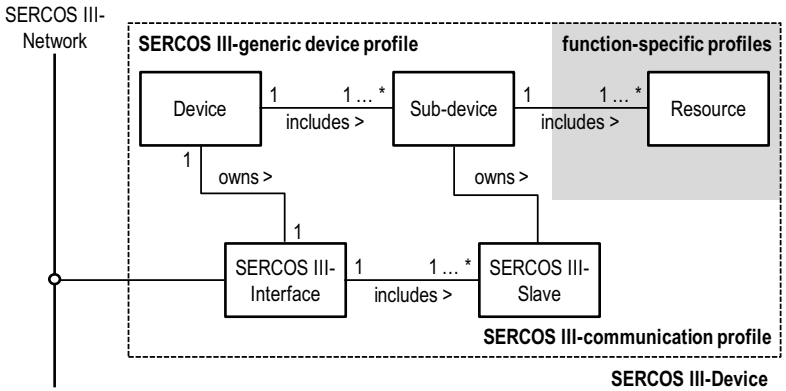


Bild 5.11: Gerätemodell von Sercos (Quelle /49/)

Bei Profdrive wird ein Gerät als Station bezeichnet (Bild 5.12). Eine Station hat genau eine Kommunikationsschnittstelle. Eine Station kann mehre Profidrive Geräte (Profidrive Device) beinhalten. Es wird lediglich die Ausprägung als Antriebseinheit (Drive Unit) betrachtet, die mehrere Antriebsobjekte (Drive Objekts) zusammenfasst. Ein Antriebsobjekt entspricht genau einem Antrieb.

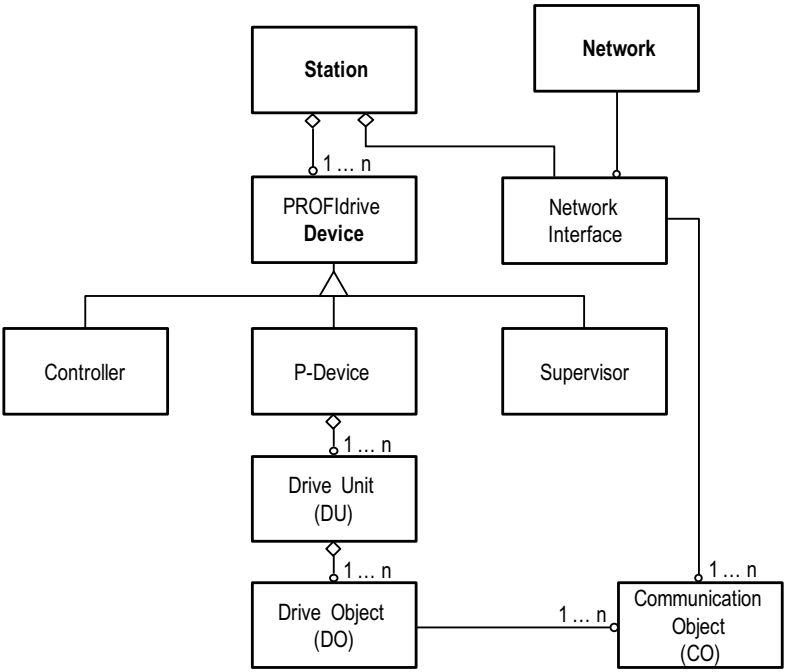


Bild 5.12: Gerätemodell von Profidrive (Quelle /49/)

Bei CIP Motion kann ein Gerät, das über ein Identity Objekt adressiert wird, ebenfalls mehrere Antriebe enthalten (Bild 5.13). Ein Identity Objekt entspricht einem Gerät.

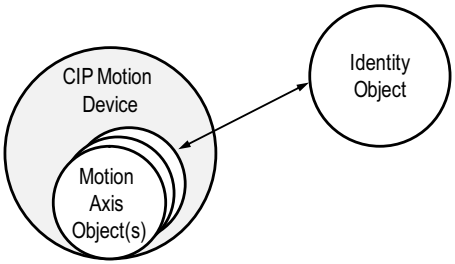


Bild 5.13: Gerätemodell für CIP Motion (Quelle /49/)

Bei CIA 402 enthält ein Gerät eine adressierbare Kommunikationsschnittstelle. Der Parameterraum des Geräts ist in Bereiche unterteilt. Es existiert keine separate Adressierung einzelner Antriebe innerhalb eines Geräts. Jeder Antrieb (axis) hat einen bestimmten Adressbereich der Parameter (Bild 5.14).

6000_h to $67FF_h$: axis 0
6800_h to $6FFF_h$: axis 1
7000_h to $77FF_h$: axis 2
7800_h to $7FFF_h$: axis 3
8000_h to $87FF_h$: axis 4
8800_h to $8FFF_h$: axis 5
9000_h to $97FF_h$: axis 6
9800_h to $9FFF_h$: axis 7

Bild 5.14: Gerätemodell für CiA 402 (Quelle /49/)

5.3.3 Antriebsprofile

Für die Definition einer funktional einheitlichen Applikationsschnittstelle für Ethernetbasierte Bussysteme sind insbesondere die Applikationsprofile zu betrachten. Hierbei werden die genormten Antriebsprofile betrachtet. Sie beschreiben die logische Sicht auf digitale Antriebe. Sie werden im Folgenden anhand der im vorhergehenden Kapitel festgelegten Funktionalitäten verglichen.

5.3.3.1 Zustandsmaschinen

Analog zum Hochlauf der Kommunikation ist zur Steuerung von digitalen Antrieben eine Zustandsmaschine beschrieben. Die jeweiligen Zustandsmaschinen der betrachteten Antriebsprofile sind in Bild 5.15 gegenübergestellt.

Durch die einzelnen Zustände wird es der Applikation auf Seiten der Steuerung ermöglicht, das Verhalten eines Antriebs zu beeinflussen. Sie ermöglichen es der Steuerungsapplikation die Zustände des Antriebs vom Einschalten bis zum gewünschten Betriebszustand, in dem der Antrieb den Sollwerten folgt, zu schalten. Ebenso kann der Antrieb auf verschiedene Weisen angehalten werden. Das Anhalten mit einer definierten Bremsrampe erfolgt durch den Übergang aus dem Betriebszustand in den vorherigen Zustand. Ein maximal schnelles Anhalten erfolgt durch das Schalten in den jeweiligen Bremszu-

stand (*Quick Stop Active*, *Shutdown*, *Switching Off*, *E-Stopp*). Zustandswechsel, die von der Steuerung aus vorgegeben werden, erfolgen durch ein Steuerwort. Jedes der betrachteten Antriebsprofile beschreibt ein Steuerwort, in dem die jeweiligen Zustände von der Steuerung an den Antrieb vorgegeben werden können.

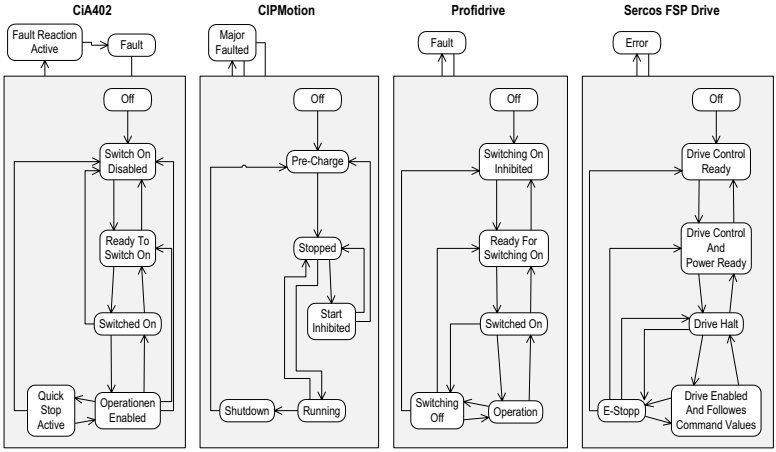


Bild 5.15: Zustandsmaschinen der betrachteten Antriebsprofile

Zusätzlich kann die Applikation auf Seiten des Antriebs interne oder externe Änderungen von Signalen durch Zustandsänderungen mitteilen. Sie kann somit Zustände abhängig von der Leistungsversorgung mitteilen. Ebenso können antriebsseitige Fehler mitgeteilt und gegebenenfalls der Antrieb auf unterschiedliche Arten angehalten werden. Den aktuellen Zustand teilt der Antrieb über ein Statuswort an die Steuerung mit.

Welche Eigenschaften die einzelnen Zustände haben, ist in Tabelle 5.5 gegenübergestellt. Dabei zeigt sich, dass sich die Zustände zwischen den Antriebsprofilen kaum unterscheiden. Lediglich bei CIP Motion existiert kein Zustand, in dem der Antrieb in Regelung ist und die aktuelle Position hält, ohne vorgegebenen Sollwerten zu folgen.

	CiA 402							CIP Motion						Profidrive						Sercos FSP Drive					
	Switch On Disabled	Ready To Switch On	Switched On	Operation Enabled	Quick Stop Active	Fault Reaction Active	Fault	Pre-Charge	Stopped	Start Inhibited	Running	Shutdown	Major Fault	Switching On Inhibited	Ready For Switching On	Switched on	Operation	Switching Off	Fault	Drive Control Ready	Drive Control and Power Ready	Drive Halt	Drive Enabled an follows Cmd.	E-Stop	Error
Steuerteil ein	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Zwischenkreis-spannung vorhanden	-	X	X	X	X	X	-	-	X	X	X	X	-	-	X	X	X	X	-	-	X	X	X	X	-
Momentenfreigabe	-	-	X	X	X	X	-	-	-	-	X	X	-	-	-	X	X	X	-	-	-	X	X	X	-
Antrieb folgt Sollwerten	-	-	-	X	-	-	-	-	-	-	X	-	-	-	-	-	X	-	-	-	-	-	X	-	-
Antrieb anhalten	-	-	-	-	X	X	X	-	-	-	-	X	X	-	-	-	-	X	X	X	-	-	-	X	X
Fehler	-	-	-	-	-	X	X	-	-	-	-	-	X	-	-	-	-	-	X	-	-	-	-	-	X

Tabelle 5.5: Übersicht der einzelnen Zustände der Antriebszustandsmaschinen

5.3.3.2 Antriebsparameter und Antriebsreglerstruktur

Der Betriebszustand, in dem der Antrieb den Sollwerten folgt, unterscheidet sich in verschieden Betriebsarten (zum Beispiel Positions- oder Geschwindigkeitsregelung). Bevor die Betriebsarten im Einzelnen gegenübergestellt werden, sollen zunächst die Antriebsparameter und Antriebsreglerstruktur verglichen werden, da sie durch die jeweiligen Betriebszustände beeinflusst werden.

In Tabelle 5.6 sind die Parameter aufgeführt, die von den jeweiligen Profilen zur Parametrierung des Antriebssystems verwendet werden, bzw. zur Diagnose abgefragt und überwacht werden können. Diese Parameter sind zunächst unabhängig von der Antriebsreglerstruktur und der aktiven Betriebsart. Sie beschreiben die Eigenschaften des Motors und des Antriebssteuerteils im Allgemeinen.

Bei diesen Parametern sind große Unterschiede zwischen den einzelnen Antriebsprofilen zu erkennen. Lediglich das Typenschild des Motors (zum Beispiel Hersteller, Seriennummer, Motortyp) wird von allen Profilen spezifiziert. Bei Profidrive sind keine weiteren Informationen über den Motor oder den Antriebverstärker, außer der Firmware, über

das Applikationsprofil beschrieben. CIP Motion geht hier bei der Beschreibung am weitesten. Dieses Antriebsprofil beschreibt Parameter für die gesamte Motorparametrierung und Diagnose. Es können Spannungen und Ströme der einzelnen Phasen vom Antrieb und Antriebsverstärker ausgelesen werden. Ebenso sind Informationen zur Induktivität und dem Drehfeld vorhanden.

	CiA 402	CIP Motion	Profidrive	Sercos FSP Drive
Typenschild	X	X	X	X
Nennspannung	-	X	-	X
Nennstrom	X	X	-	X
Nennmoment	X	X	-	X
Spitzenstrom	X	X	-	X
Nenndrehzahl	X	X	-	X
Maximaldrehzahl	X	X	-	X
Zwischenkreisspannung	X	X	-	X
Motortemperatur	-	X	-	X
Phasenwinkel	-	X	-	-
Temperatur Motorkühlung	-	-	-	X
Verstärkertemperatur	-	X	-	X
Firmwareversion	-	-	X	X
Besonderheiten	-	Parameter für Motorparametrierung und Diagnose	-	-

Tabelle 5.6: Übersicht der spezifizierten Antriebsparameter

Neben den allgemeinen Parametern zum Motor und Antriebsregler lässt sich die Struktur eines Antriebs aus Sicht der Applikation als eine Kaskade aus Positionieren, Interpolation, Positions-, Geschwindigkeits- und Momentenregelkreis darstellen (Bild 5.16). In diese Struktur lassen sich alle betrachteten Spezifikationen der Antriebsprofile abbilden. Eine strukturelle Einteilung nach Grenzwerten, Sollwerten und Reglerparametern über die Kaskaden hinweg ist nicht geeignet, da dabei die Übersichtlichkeit verloren geht und keine direkte Zuordnung zu den Betriebsarten möglich ist. Zu der Struktur nach den Kaskaden gehört ebenfalls die Betrachtung, wie ein Antrieb bei der Verwendung eines relativen Messsystems referenziert werden kann. Das Referenzieren ist notwendig für das Positionieren und Interpolieren im Antrieb.

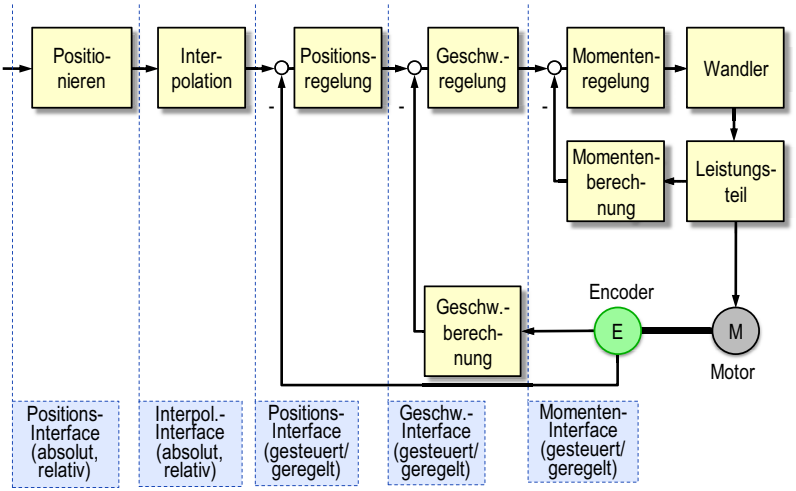


Bild 5.16: Reglerkaskaden eines digitalen Antriebsreglers

Positionieren und Interpolation

Das Positionieren und die Interpolation sind eng miteinander verbunden und werden deshalb gemeinsam betrachtet (Bild 5.17). Bei der Interpolation wird dem Antrieb eine Zielposition vorgegeben, die er ausgehend von der aktuellen Position selbstständig anfährt. Beim Positionieren können mehrere Zielpositionen an den Antrieb übergeben werden, die er dann nacheinander anfährt. Sie werden in einem Speicher abgelegt und nacheinander angefahren. Beim Positionieren sind auch unterschiedliche Übergangsbedingungen zwischen den einzelnen Positionen möglich, wie beispielsweise die Übergangsgeschwindigkeit. Sie gibt an, welche Geschwindigkeit der Motor am Übergang zwischen den beiden Positionierbefehlen haben soll. Die Interpolation ist also eine Untermenge des Positionierens. Das Positionieren ist bei CIP Motion nicht spezifiziert.

Bei der Interpolation wird von der Steuerungsapplikation direkt oder aus dem Zielspeicher des Positionierens eine Zielposition übergeben. Dazu ist festgelegt, wie der Wert zu interpretieren ist. Die Zielposition kann absolut zum Referenzpunkt oder einem Offset oder relativ zur aktuellen Position angegeben werden. Mit der Skalierung können weitere Informationen zur Position übergeben werden. Hierbei handelt es sich um die Infor-

mation, ob es sich um eine Dreh- oder Linearbewegung handelt. Auch Getriebeübersetzungen können berücksichtigt werden. Diese Funktionen zur Transformation des Sollwertes unterstützen alle betrachteten Antriebsprofile.

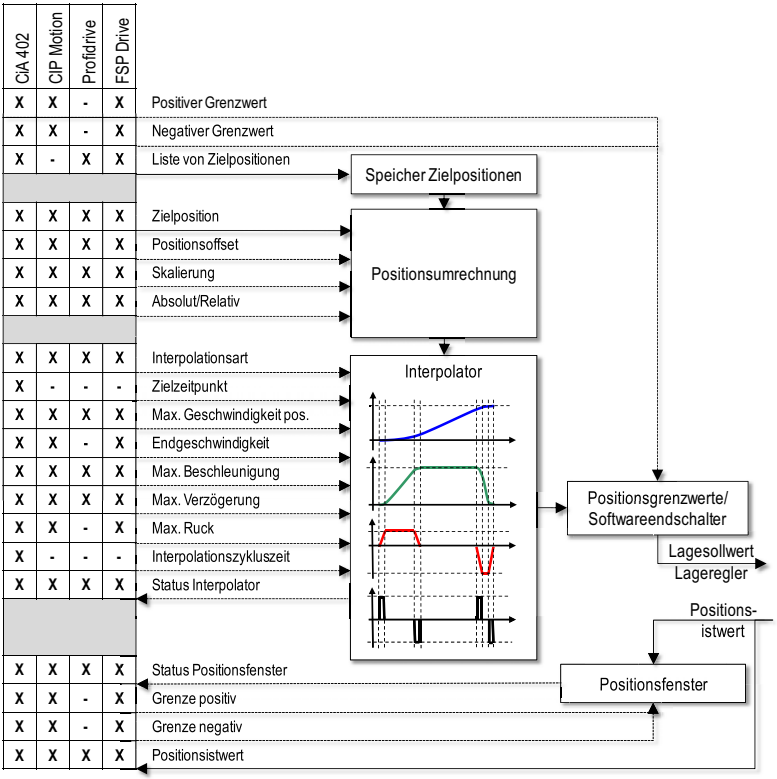


Bild 5.17: Vergleich der Funktionen für das Positionieren und Interpolieren

Über welches Profil die Sollwerte für den unterlagerten Positionsregler erzeugt werden, kann ebenfalls durch die Funktionen der Interpolationsart, wie ruckbegrenzte Interpolation oder Drehrichtung bei der Vorgabe von Winkelpositionen, vorgegeben werden. Weitere Parameter zur Begrenzung der Geschwindigkeit, Beschleunigung oder Ruck lassen ein exaktes Interpolationsprofil vorgeben. Beim Positionieren kann die Über-

gangsgeschwindigkeit zwischen den einzelnen Zielpositionen vorgegeben werden. Der aktuelle Status des Interpolators der Antriebsapplikation kann von der Steuerungsapplikation über verschiedene Status abgefragt werden. Über ein Positionsfenster kann überwacht werden, ob sich der Antrieb innerhalb eines vorgegebenen Bereichs befindet.

Bei den Funktionen des Interpolators unterscheiden sich die Profile lediglich darin, dass bei Profidrive keine ruckbegrenzte Interpolation möglich ist. CiA 402 bietet die Möglichkeit, den gewünschten Endzeitpunkt, an dem die Zielposition erreicht werden soll, vorzugeben und daraus das entsprechende Profil zu berechnen. Zusätzlich kann die Interpolationszykluszeit, in dem die Sollwerte erzeugt werden sollen, vorgegeben werden.

Die vom Interpolator berechneten Positionswerte werden auf die vorgegebenen Grenzwerte hin überprüft. Die Parameter für die Grenzwerte können außer bei Profidrive von der Steuerungsapplikation eingestellt werden.

Zur Überwachung der aktuellen Position kann die Steuerungsapplikation die aktuelle Istposition überwachen, oder aber ein Positionsfenster definieren. Befindet sich der Antrieb innerhalb des Positionsfensters, meldet er dies an die Steuerungsapplikation. Bei Profidrive ist die Rückmeldung im Statuswort definiert, die Definition der Grenzen für das Positionsfenster ist jedoch herstellerspezifisch und nicht vom Antriebsprofil spezifiziert.

Positionsregelung

Bei der Positionsregelung werden die Sollwerte entweder zyklisch aus dem Interpolator des Antriebs heraus oder aus der Steuerungsapplikation im Takt der Kommunikation an die Positionsregelung übergeben (Bild 5.18). Der Sollwert wird ebenfalls zunächst wie bei der Interpolation transformiert und überprüft, ob er innerhalb der vorgegebenen Grenzen liegt. Lediglich Profidrive beschreibt diese Grenzen nicht als Funktionen im Antriebsprofil.

Für den Positionsregler selbst können Parameter für das Regelverhalten vorgegeben werden. Der Verstärkungsfaktor für den P-Regler (Proportional) wird von allen Antriebsprofilen, außer CiA 402 beschrieben. Die Zeitkonstante für einen zusätzlichen Integralteil eines PI-Reglers (Proportional Integral) spezifizieren jedoch lediglich FSP

Drive und CIP Motion. Der Schleppabstand kann bei allen Antriebsprofilen über einen Parameter ausgelesen werden. CIP Motion lässt es zu, ein Maximum vorzugeben, das beim Überschreiten einen Fehler im Antrieb auslöst. Zur weiteren Verbesserung der Regelgüte beschreiben die Antriebsprofile CiA 402, CIP Motion und Sercos FSP Drive Parameter für den Verstärkungsfaktor zur Geschwindigkeits- und Beschleunigungsvorsteuerung.

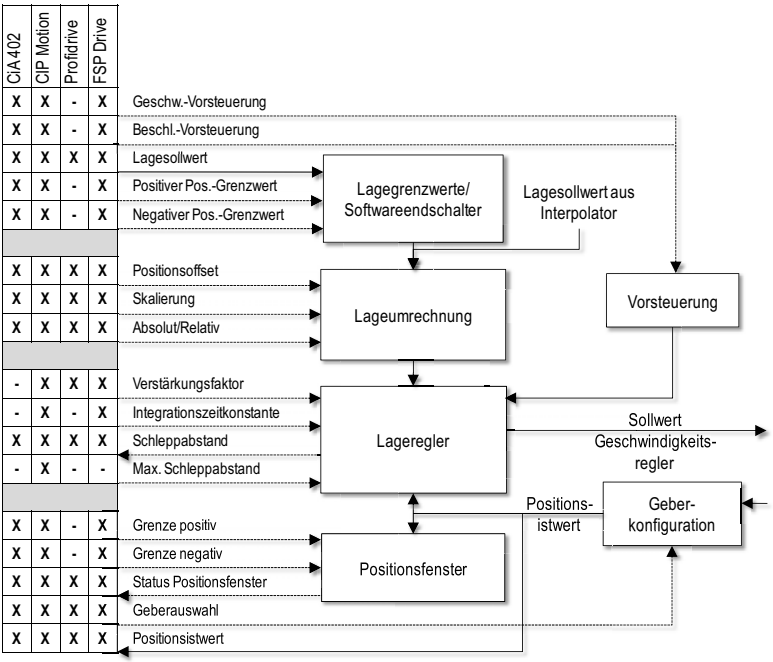


Bild 5.18: Vergleich der Funktionen des Positionsreglers

Sowohl für die Erfassung der Istposition zum Schließen des Regelkreises, als auch für die Überwachung in der Steuerungsapplikation kann jeweils der Geber konfiguriert werden. Hierbei wird festgelegt, ob der Positionswert aus dem internen Motorgeber oder einem zusätzlichen externen Positionsmesssystem erfasst werden soll.

Die Funktionalität des Positionsfensters zur Überwachung eines Positionsbereichs ist analog zu der des Positionierens und der Interpolation.

Geschwindigkeitsregler

Dem Geschwindigkeitsregler werden zyklisch Sollwerte entweder resultierend aus der Lageabweichung oder aus der Steuerungsapplikation im Kommunikationszyklustakt übergeben (Bild 5.19). Die Sollwerte werden, wie bei der Positionsregelung, entsprechend transformiert und es wird überprüft, ob die parametrierten Grenzen eingehalten werden.

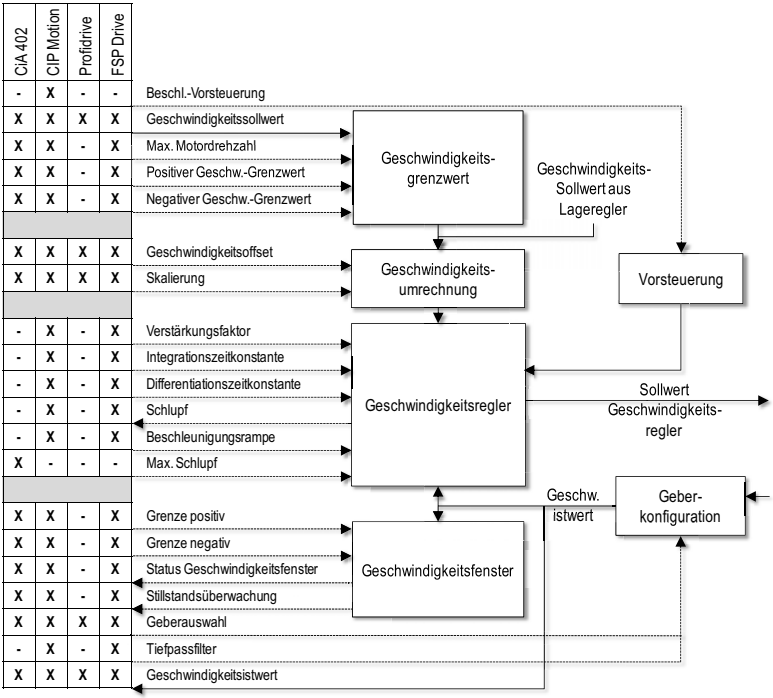


Bild 5.19: Vergleich der Funktionen des Geschwindigkeitsreglers

Das Regelverhalten kann bei CIP Motion und FSP Drive über Parameter für den Verstärkungsfaktor, die Integrations- und Differentiationszeitkonstante zur Beschreibung eines PID-Reglers (Proportional Integral Derivative) beeinflusst werden. Bei den beiden anderen Profilen sind diese Parameter herstellerspezifisch und nicht im Antriebsprofil spezifiziert. CIP Motion beschreibt für die Geschwindigkeitsregelung zusätzlich die Funktion der Beschleunigungsvorsteuerung zur weiteren Beeinflussung des Regelverhaltens.

Der Regelfehler kann als Schlupf von der Steuerungsapplikation überwacht werden. CiA 402 bietet die Möglichkeit, einen maximalen Schlupf zu definieren. Die berechnete Stellgröße wird als Sollwert an die Kaskade des Momentenreglers übergeben.

Wie auch bei der Positionsregelung lässt sich ein Fenster parametrieren, das überprüft, ob die Istgeschwindigkeit innerhalb der vorgegeben Grenzen liegt. Zusätzlich kann überwacht werden, ob der Antrieb still steht.

Die Bestimmung der Istgeschwindigkeit erfolgt durch Differentiation des Lagewertes aus dem internen Geber oder die Auswertung eines externen Geschwindigkeitssensors. Welches Geschwindigkeitssignal genutzt werden, bzw. zyklisch über die Kommunikation übertragen werden soll, kann bei allen Antriebsprofilen konfiguriert werden. Bei CIP Motion und FSP Drive existiert zusätzlich die Funktion, das differenzierte Geschwindigkeitssignal über einen Tiefpass zu filtern. Dieser kann durch Vorgabe der Zeitkonstante parametriert werden.

Momenten- und Stromregler

Die zyklischen Sollwerte für die Momentenregelung stammen entweder aus dem Geschwindigkeitsregler oder in einem sehr hohen (31,25-125 µs) Kommunikationstakt aus der Steuerungsapplikation (Bild 5.20). Der Momentensollwert wird ebenfalls durch die Vorgabe eines Offsets und einer Skalierung transformiert. Der Momentensollwert wird auf die vorgegebenen Momentengrenzen hin direkt oder auf den indirekt daraus resultierenden Motorstrom geprüft. Profidrive spezifiziert keinerlei Funktionen zur Momentenregelung. Lediglich der Momentenistwert kann über einen Parameter des Antriebsprofils abgerufen werden.

Das Regelverhalten lässt sich bei CIP Motion und FSP Drive durch die Parameter eines PI-Reglers jeweils für die Regelung des Moments und des Stroms beeinflussen.

Eine Fensterüberwachung des Istmoments wird nur vom FSP Drive spezifiziert. Die Möglichkeit zur Filterung der Momentenistwerte über einen Tiefpass oder einen Notch-Filter (schmalbandige Bandsperre). beschreibt nur CIP Motion.

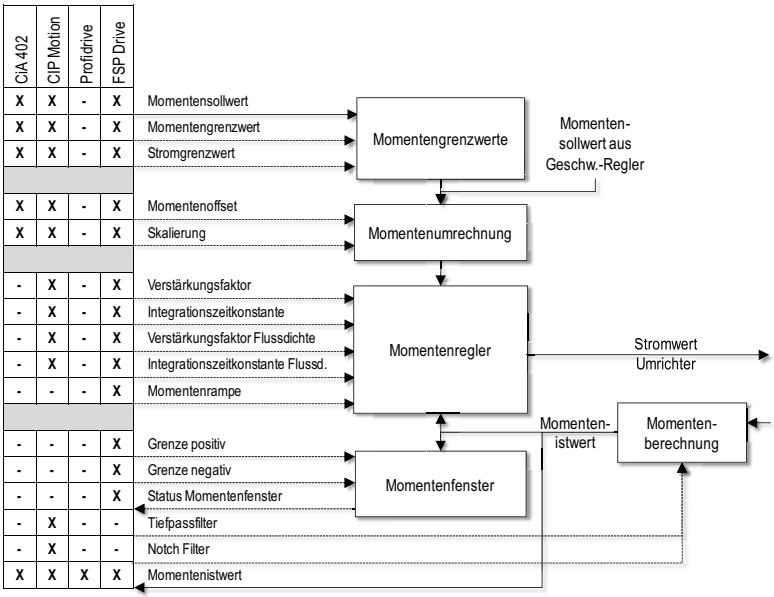


Bild 5.20: Vergleich der Funktionen des Momentenreglers

Referenzierung

Die Referenzierung eines Antriebs bei der Verwendung eines relativen Messystems oder Verschiebung des Bezugspunkts bei einem absoluten Messystem zur Positionserfassung kann auf zwei Arten erfolgen. Beim antriebsgeführten Referenzieren erzeugt der Antrieb selbst Sollwerte und fährt, bis er einen Referenzschalter, Nullimpuls oder Festanschlag erreicht. Beim steuerungsgeführten Referenzieren gibt die Steuerung Sollwerte vor und überwacht den Referenzschalter. Die jeweiligen Funktionen zum Referenzieren sind in

Tabelle 5.7 gegenübergestellt. CiA 402 und FSP Drive beschreiben das antriebsgeführte Referenzieren. Profidrive legt die Signale im Steuer- und Statuswort fest, die Umsetzung ist jedoch herstellerspezifisch. CIP Motion und FSP Drive legen das Vorgehen zum steuerungsgeführten Referenzieren durch die Steuerungsapplikation fest.

	Funktion	CiA402	CIPMotion	Profidrive	FSP Drive
Antriebsgeführtes referenzieren	Starten	X	-	X	X
	Stoppen	X	-	X	X
	Richtung	X	-	-	X
	Geschwindigkeit	X	-	-	X
	Beschleunigung	X	-	-	X
	Momentengrenzwert	-	-	-	X
	Parametrierung (Referenzschalter, Flanken)	X	-	-	X
Steuerungs-geführtes Referenzieren	Sollwerte	-	X	-	X
	Abfrage Referenzschalter	-	X	-	X
	Offsetberechnung	-	X	-	X

Tabelle 5.7: Vergleich der Funktionen zu Referenzierung

5.3.3.3 Betriebsarten

Die Betriebsarten unterscheiden sich abhängig davon, in welche Kaskade die Sollwerte von der Steuerungsapplikation in die Reglerstruktur eingebracht werden. Abhängig von den im vorhergehenden Abschnitt dargestellten Funktionen der jeweiligen Reglerkaskaden ergeben sich die unterstützten Betriebsarten nach Tabelle 5.8. Wie bereits bei den einzelnen Reglerkaskaden deutlich wurde, unterstützt CIP Motion kein Positionieren und antriebsgeführtes Referenzieren. Profidrive beschreibt keine Parameter zur Momentenregelung und beim antriebsgeführten Referenzieren ist es nur möglich, diese Funktion zu starten. Eine Parametrierung kann jedoch nicht über das Antriebsprofil erfolgen.

Die unterstützten Betriebsarten eines Antriebs werden, wie die Zustände auch, über das jeweilige Antriebssteuerwort durch die Steuerungsapplikation beauftragt. Die aktuelle Betriebsart, in der sich der Antrieb befindet, meldet er über das Statuswort. Für Be-

triebsarten, die eine zyklische Übertragung von Sollwerten erfordern, müssen die zugehörigen logischen Verbindungen konfiguriert sein. In den logischen Verbindungen der Kommunikation müssen dabei die Sollwerte für die Betriebsart enthalten sein.

	CiA402	CIPMotion	Profidrive	FSP Drive
Positionierung	X	-	X	X
Interpolation	X	X	X	X
Positionsregelung	X	X	X	X
Geschwindigkeitsregelung	X	X	X	X
Momentenregelung	X	X	-	X
Antriebsgeführtes Referenzieren	X	-	(X)	X
Steuerungsgeführtes Referenzieren	-	X	-	X

Tabelle 5.8: Übersicht der unterstützten Betriebsarten durch die Antriebsprofile

5.4 Zusammenfassung

In diesem Kapitel wurde ein detaillierter Vergleich anhand von Funktionen und der vorher definierten Struktur von Funktionalitäten durchgeführt. Die dabei identifizierten Gemeinsamkeiten und Unterschiede einzelner Kommunikations- und Applikationsprofile von Ethernet-basierten Bussystemen dienen als Grundlage zur Definition einer funktional einheitlichen Applikationsschnittstelle im nachfolgenden Kapitel.

Bei der Kommunikation sind Unterschiede im Buszugriff und Protokollaufbau festzustellen. Ethercat und Sercos III verwenden Sammeltelegramme, die vom Master erzeugt und von den Slaves durchgereicht werden. Dadurch ergeben sich auch Unterschiede in den möglichen Topologien. Weiter zeigen sich Unterschiede bei der Synchronisation. Sercos III sendet in jedem Zyklus ein Synchronisationstelegramm, Profinet und Ethercat hingegen nutzen verteilte Uhrzeiten. Die Zustandsmaschinen für den Hochlauf der Kommunikation sind nicht identisch, insbesondere die möglichen Transitionen unterscheiden sich. Die Analyse der logischen Verbindungen zeigt, dass es hier ebenfalls Gemeinsamkeiten gibt.

Als Applikationsprofile werden die vier Antriebsprofile CiA 402, CIP Motion, Profidrive und Sercos FSP Drive gegenübergestellt. Bei der Betrachtung der Antriebszustände, Parameterstruktur und den Gerätemodellen wird eine hohe Übereinstimmung festgestellt. Die Datentypen unterscheiden sich sehr stark, insbesondere bei den speziellen Definitionen für Zeiten bei Profidrive.

Bei den Antriebsprofilen werden neben kleinen Unterschieden in einigen Parametern der Reglerstruktur die unterschiedlichen Sichtweisen auf einen digitalen Antrieb deutlich. CiA 402 hat eine sehr starke Ausrichtung auf die Interpolation und das gesteuerte Verhalten eines Antriebs. Das FSP Drive hingegen legt den Fokus sehr stark auf geregelte Antriebe für Werkzeugmaschinen und Motionanwendungen. CIP Motion betrachtet einen Antrieb sehr stark aus elektrotechnischer Sicht, das sich in den vielen Parametern zu Spannungen und Strömen der Elektronik wiederspiegelt. Es werden wie beim FSP Drive viele Reglerparameter spezifiziert. Profidrive hingegen beschreibt einige Funktionen (zum Beispiel antriebsgeführtes Referenzieren), die nicht über das Profil beeinflussbar sind und die Parameter hierbei zu definieren, wird dem Hersteller überlassen.

Trotz der identifizierten Unterschiede gilt es im Folgenden, eine funktional einheitliche Applikationsschnittstelle für Ethernet-basierte Bussysteme aus dem Vergleich abzuleiten, die die Anforderungen aus Kapitel 2 erfüllt.

6 Ableiten einer funktional einheitlichen Schnittstelle für die bussystemunabhängige Kommunikation zwischen Applikationen

In diesem Kapitel wird die Ableitung einer funktional einheitlichen Applikationsschnittstelle, ausgehend vom durchgeführten Vergleich, aufgezeigt. Hierzu ist zunächst eine allgemeine Systemarchitektur zu definieren, in der sich die Schnittstelle eingliedert. Danach muss für jede der Funktionalitäten und der zugehörigen Funktionen festgelegt werden, ob sie in eine einheitliche Schnittstelle übernommen und in welcher Form sie einheitlich abgebildet werden. Diese Untersuchung erfolgt anhand der in Kapitel 4 hergeleiteten Struktur.

6.1 Allgemeine Systemarchitektur

Bevor eine konkrete Beschreibung der einzelnen Funktionalitäten einer einheitlichen Applikationsschnittstelle möglich ist, muss eine allgemeine Systemarchitektur festgelegt werden. Diese Festlegung ist notwendig, um die vereinheitlichten Funktionen in bestehende und zukünftige Automatisierungssysteme integrieren zu können. Bei der Systemarchitektur ist deshalb zwischen dem Bussystemtreiber und den Applikationsprofilen zu trennen (Bild 6.1). Die dargestellte Systemarchitektur lässt sich sowohl für einen Master, als auch für einen Slave anwenden.

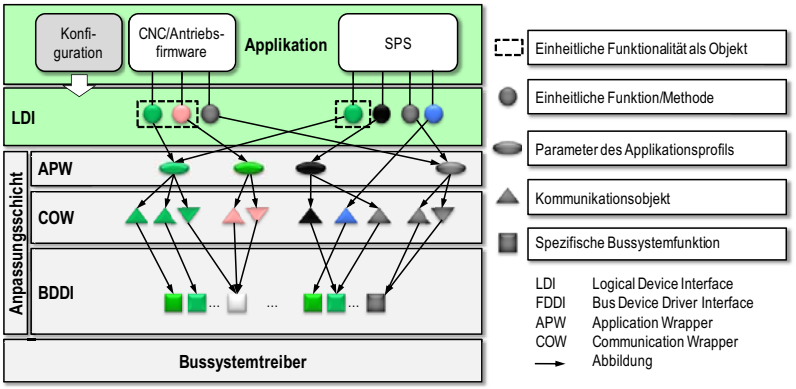


Bild 6.1: Systemarchitektur und Aufbau der funktional einheitlichen Applikationsschnittstelle und der Anpassungsschicht

Aus der Applikation erfolgt der Zugriff auf die Anpassungsschicht über die funktional einheitliche Applikationsschnittstelle. In dieser Schnittstelle werden der Applikation die Gerätefunktionen als Methoden dargestellt, die einen Zugriff auf Funktionen erlauben. Diese Funktionen lassen sich zu Funktionalitäten in Form von logischen Objekten bündeln. Diese funktional einheitliche Applikationsschnittstelle wird als logische Geräteschnittstelle (LDI - Logical Device Interface) bezeichnet.

Die Schnittstelle kann aus der Applikation heraus konfiguriert werden. Dazu zählt insbesondere die Konfiguration der logischen Verbindungen für die zyklische Kommunikation.

Die Funktionen des LDI werden in der darunterliegenden Anpassungsschicht APW (Application Wrapper) auf die spezifischen Funktionen bzw. Parameter des Applikationsprofils abgebildet. Hierzu werden im APW Regeln zur Abbildung von Funktionen des LDI auf die spezifischen Funktionen des jeweiligen Applikationsprofils hinterlegt.

Die spezifischen Funktionen werden in der COW (Communication Wrapper) Kommunikationsobjekten zugewiesen. Handelt es sich im LDI um eine Funktion der Kommunikation, wird diese direkt an auf ein Kommunikationsobjekt abgebildet.

Die Kommunikationsobjekte werden über das BDDI (Bus Device Driver Interface) an den Bussystemtreiber, der die Kommunikation abwickelt, übergeben.

Die Trennung der Anpassungsschicht in mehrere Ebenen hat den Vorteil gegenüber einer Anpassungsschicht ohne Ebenen, dass eine Trennung zwischen der Anpassung auf die Applikationsprofile und die Kommunikation besteht. Dadurch können weitere Profile über das LDI abgebildet werden und es ist nur eine Erweiterung der APW erforderlich. Zusätzlich ist es möglich, ein Profil über ein anderes Kommunikationsprotokoll durch Austausch der COW zu realisieren. Welche der Funktionalitäten und Funktionen im LDI abgebildet werden, wird im Folgenden betrachtet.

6.1.1 Entscheidungsverfahren zur Funktionsauswahl

Der Vergleich der Funktionen im vorherigen Kapitel hat gezeigt, dass es teilweise sehr große Überschneidungen bei den spezifizierten Funktionen der betrachteten Bussysteme und Applikationsprofile gibt. Bevor nun eine funktional einheitliche Applikations-

schnittstelle abgeleitet werden kann, wird zunächst ein passendes Entscheidungsverfahren ausgewählt werden. Mit dem Entscheidungsverfahren wird festgelegt, welche Funktionen in der einheitlichen Schnittstelle abgebildet werden müssen.

Entscheidungsverfahren /93, 94/ lassen sich prinzipiell in zwei Kategorien einteilen: Die Einteilung erfolgt in empirische und quantitative Entscheidungsverfahren.

Empirische Entscheidungsverfahren beziehen Beobachtungen, Experimente und Erfahrungswissen in die Entscheidung mit ein. Für die Entscheidung im Fall zur Auswahl von Funktionen müssen bestehende Applikationen untersucht und Anwender befragt werden, welche Funktionen tatsächlich verwendet werden. Ein empirisches Entscheidungsverfahren ist deshalb nicht zielführend.

Quantitative Entscheidungsverfahren legen für die Entscheidung meist mathematische Verfahren wie Stochastik, Logik oder Maximierung zugrunde. Eine Betrachtung von Extremfällen würde für die Auswahl der Funktionalitäten entweder die Integration aller Funktionen in die einheitliche Schnittstelle bedeuten oder keine. Die Übernahme von Funktionen, die in allen betrachteten Bussystemen und Applikationsprofilen enthalten sind würde Funktionen ausgrenzen, wenn sie in nur einer Spezifikation nicht enthalten sind. Eine weitere Methode zur Entscheidungshilfe aus dem quantitativen Entscheidungsverfahren stellen Entscheidungsbäume dar. Sie werden zur übersichtlichen Darstellung von formalen Regeln genutzt. Ein Entscheidungsbaum besteht aus einem Wurzelknoten und beliebig vielen Knoten, sowie mindestens zwei Blättern. Dabei stellt jeder Knoten eine logische Regel dar. Ein Blatt repräsentiert eine Antwort auf das Entscheidungsproblem. Für die Auswahl der zu vereinheitlichten Funktionalitäten und Funktionen können entsprechende Regeln definiert werden.

Der Entscheidungsbaum für die Auswahl, der in der einheitlichen Schnittstelle abzubildenden Funktionen, ist in Bild 6.2 dargestellt. Es werden Funktionen ausgewählt, die in mindestens zwei der betrachteten Profile vorhanden sind, da ihnen dadurch eine gewisse Relevanz zugeschrieben werden kann. Darüber hinaus werden Funktionen vereinheitlicht, die entweder andere Funktionen (zum Beispiel Geschwindigkeit für das antriebsgeführte Referenzieren) oder den Antrieb und damit den Prozess direkt beeinflussen (zum Beispiel Verstärkungsfaktor des Positionsregelkreises).

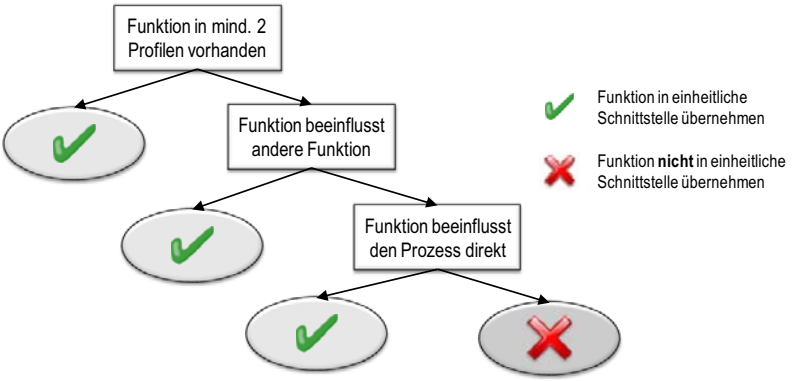


Bild 6.2: Entscheidungsbaum für die Selektion der Funktionen für die einheitliche Schnittstelle

Anhand dieses Entscheidungsbaums und des im vorherigen Kapitel durchgeführten Vergleichs werden im Folgenden die Funktionen vereinheitlicht. Sie werden nach der Struktur der Funktionalitäten aus den vorhergehenden Kapiteln betrachtet.

6.2 Topologie

Für die Verallgemeinerung der unterschiedlichen Topologie ist nur die physikalische Topologie von Interesse. Sie dient der Darstellung der Gerätetopologie innerhalb der Anlage und dient der Identifizierung und Adressierung der physikalischen Geräte. Ebenso ist sie für die Diagnose bei Problemen mit der Verkabelung (Kabelbruch) von Interesse.

Die logische Topologie, die den Informationsfluss innerhalb des Netzwerkes beschreibt, ist für die Applikation nicht von Interesse und muss deshalb nicht in einer einheitlichen Schnittstelle abgebildet werden.

Die physikalischen Topologien der betrachteten Bussysteme wie Stern, Ring, Baum, Linie lassen sich durch eine doppelt verkettete Liste darstellen (Bild 6.3). Die Darstellung erfolgt durch Zuordnung der direkt verbundenen Vorgänger und Nachfolger mit dem betrachteten Gerät. Besteht Redundanz in der Kommunikation, so hat eines der Geräte den

Kommunikationsmaster als Nachfolger. Aus diesen Informationen kann die Applikation alle möglichen physikalischen Typologien des Kommunikationsnetzwerks analysieren. Durch die doppelt verkettete Liste lassen sich Kabelbrüche bei einer physikalischen Ringstruktur einfach erkennen. Bei einer einfach verketteten Liste wären sonst zwei Listen für eine eindeutige Topologiebestimmung notwendig. Ein Kabelbruch würde bei einer einfach verketteten Liste dazu führen, dass die nachfolgenden Teilnehmer in der Liste nicht mehr erreichbar wären.

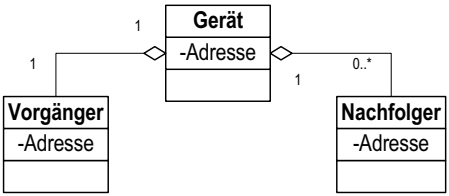


Bild 6.3: Topologiedarstellung durch doppelt verkettete Listen

6.3 Kommunikation

Im Folgenden werden die Funktionalitäten der Kommunikation betrachtet, die in der Applikationsschnittstelle verallgemeinert abgebildet werden.

6.3.1 Hochlauf

In Bild 6.4 sind die einzelnen Kommunikationszustände des Hochlaufs der betrachteten Ethernet-basierten Bussysteme gegenübergestellt. Dabei lassen sich fünf einheitliche Zustände ableiten, die für eine einheitliche Schnittstelle relevant sind.

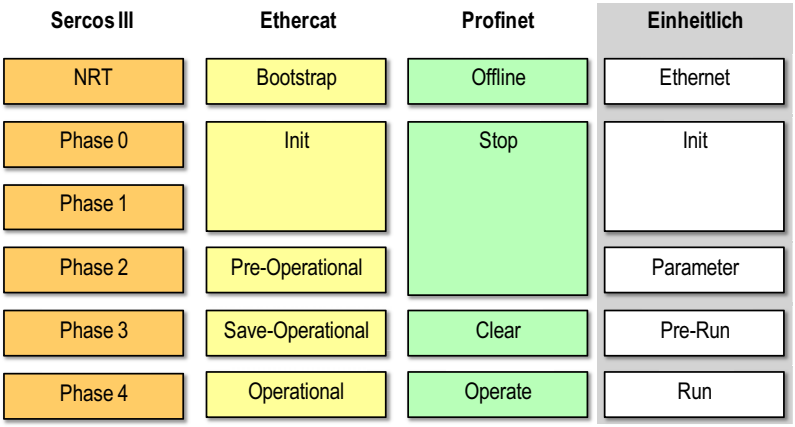


Bild 6.4: Einheitliche Zustände der Kommunikation für den Hochlauf

Die einheitlichen Zustände zeichnen sich im Einzelnen durch folgende Eigenschaften aus:

Ethernet
Zustand für normalen Dateitransfer. Es können dabei Dateien für Firmwareupdates oder ähnlichem an die Slaves übertragen werden. Der Datenaustausch erfolgt über Standard-Ethernet.

Init
Initialzustand nach dem Einschalten. Es können grundlegende Einstellungen vorgenommen bzw. ermittelt werden (Topologieerkennung, Adressvergabe,…).

Parameter
Parametriermodus. Die azyklische Kommunikation ist aktiv und Parameter für die zyklische Kommunikation können übertragen werden. Funktionen der Applikationsprofile können angesprochen und Funktionalitäten damit konfiguriert und parametriert werden.

PreRun
Vor- oder Testbetriebsmodus. Die zyklische Kommunikation ist aktiv, aber die übertragenen Daten werden von den Applikationen nicht ausgewertet. Probleme beim Echtzeit-

verhalten können in diesem Zustand festgestellt werden, ohne den Prozess zu beeinflussen.

Run
Betriebsmodus. In diesem Modus kann die Freigabe der Slaves erfolgen und die normale Prozessdatenkommunikation stattfinden. Die Soll- und Istwerte werden von den Applikationen ausgewertet.

Die jeweiligen Transitionen der betrachteten Bussysteme zwischen den einzelnen Zuständen sind extrem unterschiedlich. Aus Sicht der Applikation spielt dies jedoch keine Rolle. Die Applikation im Master gibt lediglich den gewünschten Zielzustand vor. Die Applikation im Slave erhält diesen über den Bus. Welche Zwischenzustände dabei durchlaufen werden müssen, hat keinen Einfluss auf das Prozessverhalten. Das Prozessverhalten wird nur beim Verlassen des Zustandes Run beeinflusst, da dann automatisch die Antriebsfreigabe gelöscht wird. Dieses Verhalten ist bei allen Bussystemen gleich. Somit lässt sich die in Bild 6.5 dargestellte Zustandsmaschine ableiten. Über die einheitliche Schnittstelle wird lediglich der einzunehmende Zielzustand übertragen.

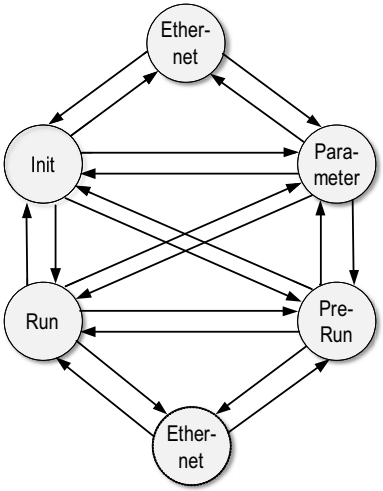


Bild 6.5: Zustände und Transitionen der einheitlichen Kommunikationszustandsmaschine

6.3.2 Buszugriff und Protokollaufbau

Die betrachteten Ethernet-basierten Bussysteme unterscheiden sich beim Buszugriff und Protokollaufbau, wie es im vorhergehenden Kapitel beschrieben wurde. Aus Sicht der Applikation ist dies jedoch nicht direkt relevant. Lediglich die Folgen daraus müssen in der Applikation bekannt sein. Ein Beispiel hierfür ist, dass Aufgrund des unterschiedlichen Protokollaufbaus bei einer großen Anzahl Kommunikationsteilnehmer die gewünschte Zykluszeit einer logischen Verbindung nicht mehr erreicht werden kann. Der Protokollaufbau und Buszugriff kann unabhängig vom jeweiligen Bustreiber autark umgesetzt werden.

6.3.3 Synchronisation

Die Synchronisation zwischen den einzelnen Kommunikationsteilnehmern muss ebenfalls in der funktional einheitlichen Applikationsschnittstelle abgebildet werden. Für den zyklischen Datenaustausch hat über diese Schnittstelle eine Synchronisation mit dem Bussystemtreiber und damit den verteilten Applikationen des Masters und den Slaves zu erfolgen. Die Schnittstelle muss für jede der logischen Verbindungen eine Zykluszeit festlegen, in der die Daten ausgetauscht werden. Die Applikation muss sicherstellen, dass innerhalb dieses Zyklus neue Daten zum Senden verfügbar sind und die empfangenen Daten verarbeitet werden. Die Synchronisation über die Schnittstelle erfolgt über eine gemeinsame Zeitbasis. Somit kann für den Sollwert übergeben werden, wann er gültig sein soll und der Istwert enthält die Information, wann er erfasst wurde. Die dazu erforderliche Synchronisation über das Bussystem wird von der Anpassungsschicht unterhalb des LDI übernommen. Das LDI tauscht die Daten über das entsprechende Kommunikationsobjekt und dem BDDI mit dem Bussystemtreiber aus. Die Synchronisation im BDDI zum Bussystemtreiber erfolgt mit Hilfe eines Interrupts. Der Interrupt gibt an, sobald neue Sollwerte innerhalb des Kommunikationszyklus zu schreibenbzw. zu lesen sind.

6.3.4 Datenaustausch

Die einheitliche Applikationsschnittstelle benötigt Mechanismen analog zum Datenaustausch der Bussysteme. Dazu zählen der zyklische und azyklische Datenaustausch.

Der zyklische Datenaustausch wird für jeden Kommunikationsteilnehmer separat über die logischen Verbindungen konfiguriert. Es können hierzu Funktionen bzw. Parameter aus dem einheitlichen Applikationsprofil übertragen werden. Die zyklische Verbindung tauscht die Daten nach Bild 6.6. aus. Es können mehrere Verbindungen zwischen einzelnen Geräten existieren.

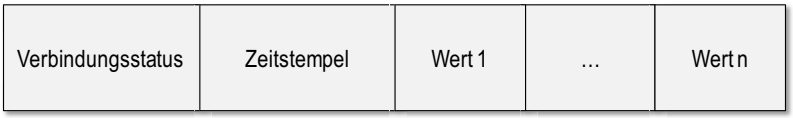


Bild 6.6: Zyklischer Datenaustausch der Applikationsschnittstelle

Dabei wird ein Verbindungsstatus ausgetauscht, der beschreibt, ob der Wert gültig ist, oder ein Fehler aufgetreten ist. Die Verbindungen können abhängig von der Betriebsart, während der Zustand Run aktiv ist, über den Verbindungsstatus aktiviert oder deaktiviert werden. Ein Zeitstempel in Anlehnung an das IEEE 1588 Datenformat gibt an, wann ein Sollwert gültig werden soll, bzw. wann ein Istwert erfasst wurde. Der eigentliche Wert stellt das Datum der Funktion, bzw. des Parameters dar. Beispiele für den Wert sind der Positionssollwert oder der Antriebsstatus.

Der azyklische Datenaustausch erfolgt über die in Bild 6.7 dargestellte Struktur, die für jeden adressierbaren Teilnehmer zur Verfügung steht. Der Inhalt des azyklischen Bereichs besteht aus einer ID zur Kennung und der Zugriffsmethode Lesen oder Schreiben. Über die ID kann die Antwort dem ursprünglichen Aufruf zugeordnet werden. Der Parameter gibt an, welche Funktion davon betroffen ist. Im Wert ist entweder das bei einem Schreibzugriff zu übertragene Datum enthalten oder die Antwort bei einem Lesezugriff. Treten Fehler auf, werden sie ebenfalls im Wert übertragen und im Zugriffsfeld markiert.

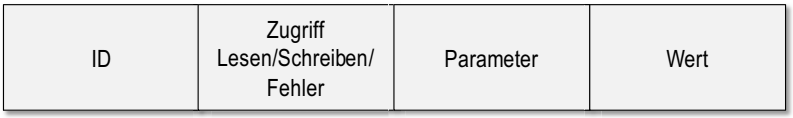


Bild 6.7: Schnittstellenbeschreibung für den azyklischen Datenaustausch

6.3.5 Logische Verbindungen

Für den zyklischen Datenaustausch werden logische Verbindungen zwischen den einzelnen Geräten beschrieben. Dabei muss jede Verbindung einzeln konfiguriert werden. Die Konfiguration wird gemäß der Beschreibung in Bild 6.8 vorgenommen. Die Verbindungen können durch die Nummerierung eindeutig zugeordnet werden. Der Offset bezieht sich dabei auf die Position innerhalb eines gemeinsamen Speichers oder Objekts, an dem die Daten der Verbindung stehen sollen. Die beiden Adressbereiche geben die Adressen der Verbindungsenden an. Die Adressen sind entsprechend dem Gerätemodell (siehe 6.4.1) zu verwenden. Danach folgt eine Liste der Parameter, die zwischen den beiden Verbindungsenden ausgetauscht werden sollen. Für jede Verbindung können unterschiedliche Zykluszeiten angegeben werden, in denen sie aktualisiert werden. Somit ist es möglich, beispielsweise Verbindungen zu E/As langsamer zu konfigurieren, als zu momentengeregelten Achsen. Die Zykluszeit muss ein ganzzahliges Vielfaches der Buszykluszeit sein. Über das Attribut aktiv/inaktiv kann festgelegt werden, ob die Verbindung verwendet werden soll oder ob sie erst später, wenn ein neues Gerät über Hotplug hinzugefügt wird, aktiviert werden soll.

Bild 6.8: Struktur der Konfiguration für logische Verbindungen innerhalb der Applikationsschnittstelle

6.4 Applikationsprofil

Eine einheitliche Sicht auf die Applikationsprofile wird entsprechend der Struktur aus dem vorhergehenden Kapitel abgeleitet. Dazu werden zunächst die einheitlichen Geräte- und Parametermodelle hergeleitet und anschließend die Antriebsprofile vereinheitlicht.

6.4.1 Gerätemodell

Die betrachteten Applikationsprofile beschreiben jeweils unterschiedliche Gerätemodelle. Ein einheitliches Gerätemodell muss sich auf die einzelnen spezifischen Gerätemodelle eindeutig abbilden lassen. Nur so ist sichergestellt, dass alle Geräte adressiert und

eindeutig zugeordnet werden können. Zur einheitlichen Beschreibung von logisch unterteilten oder modularen Geräten eignet sich das in Bild 6.9 dargestellte Gerätemodell. Dabei besteht ein Gerät aus mehreren Modulen. Ein Modul kann in weitere Elemente unterteilt werden. Drei Ebenen sind dabei ausreichend, um eine eindeutige Abbildung auf die profilspezifischen Gerätemodelle zu ermöglichen, da sie ebenfalls jeweils nur drei Ebenen beschreiben. Die Möglichkeit zur Gruppierung mehrerer Drive Objects zu einer Drive Unit bei Profidrive muss nicht in ein einheitliches Gerätemodell übernommen werden, da die Parameter der Drive Unit auch über das Drive Object adressierbar sind. Die Instanz im FSP Drive ist notwendig für die Adressierung modularer Geräte, weshalb zwei Ebenen für ein vereinheitlichtes Gerätemodell nicht ausreichend sind.

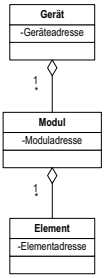
CiA402	CIPMotion	Profidrive	Sercos FSP Drive	Einheitliches Modell	Einheitliche Adresse
Device	Device	Station	Interface		IP-Adresse
		(Drive Unit)			
Adress-bereich		Drive-Object	SERCOS Slave		16 Bit Integer
			Instanz		16 Bit Integer

Bild 6.9: Einheitliches Gerätemodell zur Beschreibung von modularen Geräten

Im Detail beschreibt sich die Struktur wie folgt:

Gerät
Stellt das physikalische Gerät dar, das über einen Netzwerkanschluss verfügt. Die Geräteadresse stellt die IP-Adresse dar.

Modul
Das Modul beschreibt logische Einheiten innerhalb des Geräts. Ein Beispiel hierfür ist eine Achse innerhalb eines mehrachsigen Antriebsverstärkers. Ein Modul besitzt Parameter, die über die Kommunikation angesprochen werden können.

Element
Das Element stellt die kleinste adressierbare Einheit dar. Ein Beispiel hierfür ist eine Scheibe eines modularen E/As. Ein Element besitzt eigene Parameter, die über die Kommunikation angesprochen werden können.

6.4.1.1 Parameter

Einem Modul und einem Element des vereinheitlichten Gerätemodells können Parameter zugeordnet werden. Ein Parameter gliedert sich, wie bereits im vorhergehenden Kapitel hergeleitet, in die nachfolgende Struktur. Dabei lässt sich ein Parameter verallgemeinert darstellen.

Name

Für den Namen wird ein String mit variabler Länge für die einheitliche Applikationsschnittstelle festgelegt.

Nummer

Die Parameternummer und deren Struktur können bei der Konzeption einer einheitlichen Schnittstelle frei gewählt werden. Sie können von der Anpassungsschicht auf die profilspezifischen Parameter umgesetzt werden. Sie müssen jedoch neu definiert werden, da keines der betrachteten Applikationsprofile die Obermenge der anderen Profile darstellt.

Um der Anforderung für die Erweiterbarkeit und den Durchgriff auf busspezifische Funktionen über dieselbe Schnittstelle zu erlauben, muss sich die Parameternummer auf die busspezifische Struktur der jeweiligen Profile abbilden lassen.

In Bild 6.10 ist eine Struktur für die Nummerierung dargestellt, die es erlaubt, sowohl einheitliche als auch applikationsspezifische Funktionen über dieselbe Methode anzusprechen. Mit dieser einheitlichen Parameternummerierung ist es möglich, Funktionalitäten einem Parameter zuzuweisen und diesen in weitere Elemente für die einzelnen Funktionen zu strukturieren. Dabei ist die Struktur so gewählt, dass zum einen eine Anzahl von 65536 (2 Byte) Parameter mit jeweils ebenso vielen Elementen adressierbar sind. Dies ist für die vereinheitlichten Parameter ausreichend. Zum anderen können beim Zugriff auf bus- und applikationsspezifische Parameter die Strukturen der einzelnen

Applikationsprofile abgebildet werden. Die 2 Byte für die Parameternummer und die Elementnummer reichen aus, um busspezifische Parameter über diese Schnittstelle ansprechen zu können, wie der Vergleich im vorherigen Kapitel zeigt.

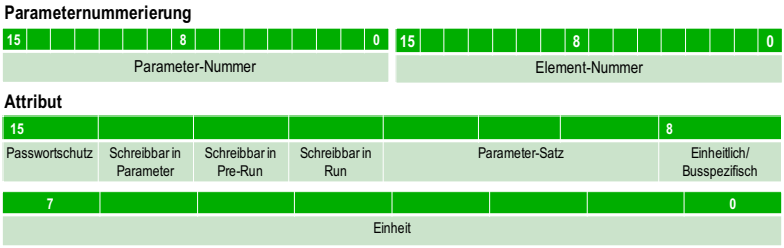


Bild 6.10: Einheitliche Struktur zur Nummerierung der Parameter für Funktionen

Attribut

Ob es sich um einen einheitlichen oder busspezifischen Parameter handelt, wird mit Bit 0 des Attributs (Bild 6.10) festgelegt. Damit wird den Anpassungsschichten APW und COW mitgeteilt, wenn es sich um einen vereinheitlichen Parameter handelt. Mit dieser Information wird in der Anpassungsschicht im Falle eines vereinheitlichten Parameters die Übersetzung auf einen busspezifischen vorgenommen. Im Falle eines busspezifischen Parameters reicht ein reines Kopieren in der Anpassungsschicht aus.

Das Attribut wird beim Datenaustausch weder zyklisch noch azyklisch übertragen. Es kann jedoch über die spezielle azyklischen Mechanismen ausgelesen bzw. konfiguriert werden.

Das Attribut beinhaltet zusätzlich zu den bereits beschriebenen Informationen weitere Angaben über die Zugriffsrechte des jeweiligen Parameters. Es enthält Informationen, in welchem Zustand des Hochlaufs ein Parameter schreibbar ist oder ob er mit einem entsprechenden Passwort schreibbar ist. Dabei kann eine Applikation prüfen, ob ein Wechsel des Kommunikationszustandes dazu führt, dass ein Parameter schreibbar ist. Beispielsweise der Wechsel vom Kommunikationszustand Run in den Zustand Parameter hat zur Folge, dass der Parameter, der angibt auf welchen Positionswert (internes oder externes Lagemesssystem) geregelt werden soll, schreibbar ist.

Das Sercos FSP Drive bietet die Möglichkeit zusätzlich zu den Parametern sogenannte Parametersätze zu verwalten. Jedem Parameter können acht verschiedene Werte zugewiesen werden, die dann zentral durch Umschalten der Parametersätze aktiviert werden. Um diese Möglichkeit für das FSP Drive über die einheitliche Schnittstelle nutzen zu können, sind 3 Bits für die acht Parametersätze vorgesehen.

Auf die Einheit im Attribut wird im Folgenden bei der einheitlichen Darstellung der Werte und Datentypen eingegangen.

Wert und Datentyp

Bei den Datentypen gibt es für die Abbildung, über eine einheitliche Schnittstelle, mehrere Möglichkeiten. Es ist denkbar, 64 Bit Datentypen zu verwenden und kleine Einheiten für jeden Parameter zu definieren. Für Längen wären damit beispielsweise bei einer Auflösung von 10^{-12} (Pikometer) ein Wertebereich von +/-9223,4 km möglich. Dies ist auch für andere physikalische Einheiten ausreichend.

Problematisch ist hierbei jedoch die Datenverarbeitung (Adressierung, Speicherplatz, Rechenoperationen). Steuerungsseitig gibt es bisher kaum Systeme, die mit diesen 64Bit Datentypen rechnen können. Slaveseitig kommen meist Mikrocontroller oder FPGAs (Field Programmable Gate Array) zum Einsatz, die ebenfalls nicht mit diesen Datentypen rechnen können. Um eine Hardware- und Geräteunabhängigkeit zu gewährleisten, wird hier eine Struktur definiert, die es erlaubt, für jeden Parameter einen geeigneten Wertebereich vorzugeben. Auch busspezifische Parameter lassen sich damit direkt ansprechen.

Die physikalische Einheit ist für jeden Parameter im Attribut hinterlegt. Es können somit 256 physikalische Einheiten angegeben werden. Dies ist ausreichend, um die SI-Einheiten (Système International d'unités) /95/ abzudecken. Darüber hinaus können weitere internationale Einheiten, wie beispielsweise Inch oder Pfund, abgebildet werden. Die unterschiedlichen Datentypen für Zeiten von Profidrive lassen sich alle auf die Datentypen der IEEE 1588 für eine Darstellung der absoluten Zeit in Nanosekunden und Sekunden vereinheitlichen.

Der Wert des Parameters wird auf eine Länge von 4 Byte festgelegt und um 1 Byte für den vorzeichenbehafteten Exponenten erweitert (Bild 6.10). Der Exponent ermöglicht

somit, die Auflösung im Bereich von $10^{+/-127}$ zu variieren. Der Exponent wird im Gegensatz zum Attribut über die Schnittstelle mitübertragen und die Auflösung kann dadurch zur Laufzeit geändert werden. Der Datentyp für die 4 Byte selbst wird für jeden Parameter fest vorgeschrieben. Sollte der Wertebereich für einen Parameter nicht ausreichen oder wird auf einen busspezifischen 64 Byte Parameter zugegriffen, muss dieser Wert auf zwei Parameter aufgeteilt werden. In diesem Fall ist eine Anpassung durch den APW erforderlich.

Bild 6.11: Struktur des Werts und des zugehörigen Exponenten

Durch die Verwendung des einheitlichen Exponenten und den Datentypen sind keine 8 Byte Datentypen für vereinheitlichte Funktionen notwendig. Dadurch wird die Flexibilität und Handhabbarkeit weiter erhöht. Der rechentechnische Aufwand für die Anpassungsschicht den Wert, die Einheiten und den Exponenten dieser einheitlichen Struktur auf die busspezifischen Datentypen anzupassen, ist deutlich geringer, als mit 8 Byte Datentypen zu arbeiten.

Default

Ein Defaultwert wird in der einheitlichen Schnittstelle nicht berücksichtigt, da dies lediglich von dem Applikationsprofil CiA 402 spezifiziert wird.

Beschreibung

Eine Beschreibung der Parameter kann über die einheitliche Schnittstelle abgefragt werden. Hierbei wird die Beschreibung aus der Spezifikation der einheitlichen Applikationsschnittstelle übernommen. Bei spezifischen Parametern wird die Beschreibung des Parameters aus dem Gerät übernommen, falls sie vorhanden ist.

Grenzwert

Für jeden Parameter kann ein oberer und unterer Grenzwert entsprechend der Einheit des Parameters im Attribut und dessen Datentyp angegeben werden. Ist kein Wert angegeben, ist der maximale Wertebereich des Datentyps abhängig vom Exponenten möglich.

Verfügbare Parameter

Welche der funktional einheitlichen Parameter verfügbar sind und auch von der Anpassungsschicht abgebildet werden können, kann über die einheitliche Applikationsschnittstelle abgerufen werden. Es werden die Parameternummern und Elemente mit ihrem Attribut aufgelistet. Somit ist auch eine Unterscheidung möglich, ob es sich um vereinheitlichte oder busspezifische Parameter handelt.

6.4.2 Antriebsprofil

Im folgenden Abschnitt soll explizit die Sicht auf elektrische und insbesondere digitale Antriebe, ausgehend vom Vergleich der Antriebsprofile aus dem vorhergehenden Kapitel, abgeleitet werden.

6.4.2.1 Zustandsmaschine

Ausgehend vom Vergleich der einzelnen Zustandsmaschinen für elektrische Antriebe lässt sich diese einheitlich, wie in Bild 6.12 dargestellt, beschreiben. Ausgehend vom Zustand nach dem Einschalten kann der Antrieb bis in den Zustand *Operation* gebracht werden.

Im Zustand *Switched Off* ist die Spannungsversorgung am Antriebsverstärker des Antriebs vorhanden und es kann mit ihm kommuniziert werden. Der Zustand *Enabled* beschreibt, dass die Leistungsversorgung für den Motor vorhanden ist und die Zwischenkreisspannung aufgebaut werden kann. Die Spannung im Zwischenkreis ist im Zustand *Switched On* vorhanden und der Antrieb befindet sich in Regelung, folgt jedoch nicht den Sollwerten. Die Sollwerte werden erst im Zustand *Operation* am Motor umgesetzt.

Es gibt vier verschiedene Zustände, die zum Anhalten des Antriebs führen. Dies kann durch einen *Quick Stop* erfolgen, bei dem mit maximaler Verzögerung (Bremswider-

stand und mechanische Bremse) der Antrieb angehalten wird. Beim Wechsel aus *Operation* in den Zustand *Switched On* wird der Antrieb nach einer Bremsrampe angehalten und hält die Position bei freigegebenem Antrieb. Beim *Coast Stop* wird der Antrieb stromlos geschaltet und trudelt aus. *Ramp Down* hält den Antrieb nach einer konfigurierbaren interpolierten Bremsrampe an und entzieht die Leistungsfreigabe. Tritt ein Fehler auf, nimmt der Antrieb, ausgehend aus jedem Zustand, immer den Zustand *Error* ein. Befindet er sich in *Operation*, wird automatisch über den *Quick Stop* in den Fehlerzustand gewechselt.

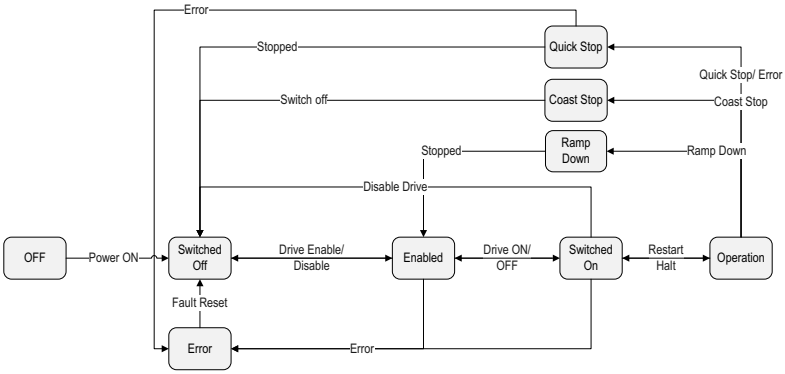


Bild 6.12: Einheitliche Zustandsmaschine für elektrische Antriebe

Diese einheitliche Zustandsmaschine für digitale Antriebe lässt sich auf die jeweiligen Zustandsmaschinen der Antriebsprofile abbilden, wie bereits der Vergleich in 5.3.3.1 zeigt. Die Transitionen können teilweise wie bei dem Hochlauf der Kommunikation nicht direkt zugeordnet werden. Sind aber zum Erreichen eines Zustands mehrere Zustandswechsel der profilspezifischen Zustandsmaschinen notwendig, kann dies ohne Beeinflussung des Prozesses oder Maschinenverhaltens durchgeführt werden.

Die Beauftragung bzw. die Statusmeldung der einzelnen Zustände sind durch einheitliche Steuer- und Statuswörter über die Schnittstelle abgebildet (Tabelle 6.1). Die Steuer und Statuswörter müssen für die zyklische Kommunikation, wie alle anderen Parameter mit Hilfe der logischen Verbindungen konfiguriert werden. Sie enthalten auch die Informationen zu den vereinheitlichten Betriebsarten, die im Folgenden anhand der Reglerstruktur hergeleitet werden.

Steuerwort																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1
Switched On	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0
Enabled	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1
Switched Off	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0
Coast Stop	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0
Quick Stop	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1
Ramp Stop	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0
Error Reset	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1
Betriebsarten	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	x	x	x

Statuswort																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Operation	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	1
Switched On	x	x	x	x	x	x	x	x	x	x	x	x	x	0	1	0
Enabled	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	1
Switched Off	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0
Coast	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0
Quick Stop	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	1
Ramp Stop	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0
Error	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	1
Betriebsarten	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	0/1	x	x	x

Tabelle 6.1: Einheitliches Steuer- und Statuswort für elektrische Antriebe

6.4.2.2 Reglerstruktur

Für die Reglerparametrierung und die Betriebsarten wird eine einheitliche Reglerstruktur festgelegt, die eine einheitliche Sichtweise auf einen elektrischen Antrieb darstellt. Hierdurch wird die geforderte Eindeutigkeit erreicht. Als Grundlage für diese Reglerstruktur dient der detaillierte Vergleich von Betriebsarten und Reglerstruktur aus dem vorhergehenden Kapitel, der anhand von Beschreibungen aus den Spezifikationen der Antriebsprofile durchgeführt wurde. In der Reglerstruktur sind Funktionen enthalten, die nach dem Entscheidungsverfahren aus 6.1.1 als relevant angesehen werden. Bild 6.13 zeigt das Ergebnis der einheitlichen Reglerstruktur.

Die Reglerstruktur gliedert sich in mehrere Kaskaden (Positionierung, Interpolation, Positions-, Geschwindigkeits- und Momentenregelung). Sie kann abhängig von der konfigurierten Betriebsart an den verschiedenen Kaskaden aufgetrennt und mit Sollwerten beauftragt werden.

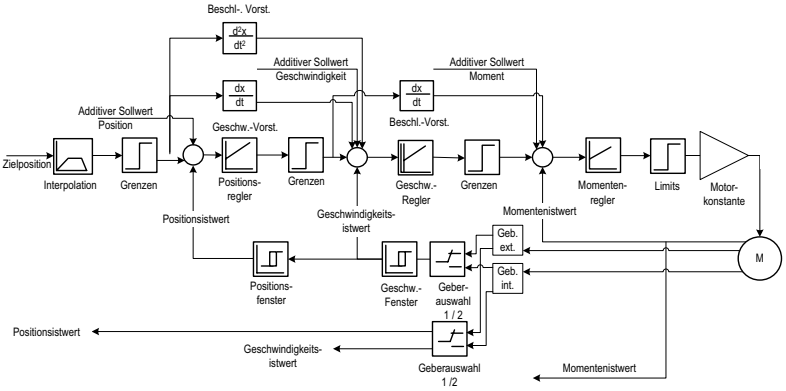


Bild 6.13: Einheitliche Reglerstruktur für elektrische Antriebe

Die Reglerstruktur beinhaltet die Interpolation bzw. das Positionieren auf dem Antrieb, Dabei berechnet der Antrieb die Sollwerte für den Positionsregler. Der Positionsregler besteht aus einem PI-Regler und einem Positionsfenster zur Istwertüberwachung. Zusätzlich existiert eine Geschwindigkeits- und Beschleunigungsvorsteuerung zur Erhöhung der Regelgüte. Der Geschwindigkeitsregler ist durch einen PID-Regler abgebildet und verfügt ebenfalls über ein Fenster zur Überwachung der Istgeschwindigkeit. Zusätzlich ist eine Beschleunigungsvorsteuerung vorhanden, die es erlaubt, den Momentensollwert zusätzlich zu beeinflussen. Für die Positionsregelung sowie die Geschwindigkeitsregelung kann konfiguriert werden, ob der Istwert aus dem internen oder externen Geber für die Regelung bezogen werden soll. Die Momenten- und Stromregelung erfolgt durch einen PI-Regler. In jeder der Kaskaden kann ein additiver Sollwert vorgegeben werden und es erfolgt eine Überwachung der Grenzwerte. Der additive Sollwert kann beispielsweise für weitere Vorsteuerungsverfahren aus der Steuerung genutzt werden. Die einzelnen Istwerte können unabhängig von der verwendeten Betriebsart zyklisch übertragen werden.

6.4.2.3 Betriebsarten

Im Folgenden werden die Betriebsarten und deren Funktionen im Detail dargestellt. Sie lassen sich ausgehend vom Vergleich der Antriebsprofile im vorhergehenden Kapitel ableiten. Parameter, die sich nicht direkt auf alle Antriebsprofile abbilden lassen und trotz-

dem enthalten sind, wurden aufgrund der Ergebnisse des beschriebenen Entscheidungsbaums mit aufgenommen.

Positionieren/Interpolation

Die Betriebsart Positionieren, bzw. Interpolation und deren Funktionsstruktur aus einheitlichen Parametern sind in Bild 6.14 dargestellt. Die Betriebsart wird über das Steuerwort aktiviert. Die logischen Verbindungen müssen entsprechend der Betriebsart konfiguriert und aktiviert sein. Das Positionieren unterscheidet sich von der reinen Interpolation nur darin, ob eine Zielposition in der Liste von Zielpositionen enthalten ist oder mehrere. Über ein Steuerbit kann die Berechnung im Interpolator gestartet werden und der Antrieb beginnt den interpolierten Werten zu Folgen. Die Interpolationsart legt fest, wie das Positionsprofil berechnet wird. Es kann beispielsweise trapezförmig oder mit einer $\sin^2$-förmigen Ruckbegrenzung berechnet werden. Beim Positionieren beschreibt die Interpolationsart zusätzlich die Geschwindigkeitsübergänge beim Wechsel zum nächsten Positionswert aus der Liste. Es kann vorgegeben werden, ob die Geschwindigkeit null oder die Geschwindigkeit einen der beiden Werte aus benachbarten Listeneinträgen annehmen soll.

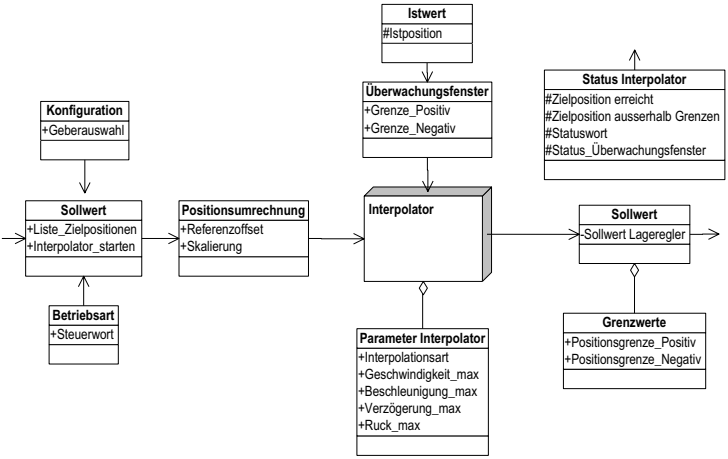


Bild 6.14: Betriebsart Positionieren/Interpolation und deren Struktur mit einheitlichen Parametern

Es besteht kein weiterer Unterschied zwischen der Interpolation und dem Positionieren. In beiden Betriebsarten kann konfiguriert werden, von welcher Position aus (aktuelle Istposition oder zuletzt interpolierte Position) interpoliert werden soll.

Die Zielposition wird abhäng von der Skalierung und dem Referenzoffset berechnet. Zusätzlich wird bei der Berechnung der Zielposition berücksichtigt, ob sie absolut oder relativ vorgegeben wird.

Der Interpolator berechnet mit den vorgegebenen Maximalwerten für Geschwindigkeit, Beschleunigung, Verzögerung und Ruck das Interpolationsprofil. Einzelne Interpolationspunkte werden nach Überprüfung der Positionsgrenzwerte im Positionsregeltakt an die unterlagerte Kaskade übergeben.

Der Positionsistwert kann für verschiedene Geber (interner oder externer Geber) überprüft werden. Bei der Überprüfung kann ermittelt werden, ob die Zielposition erreicht ist oder der Istwert innerhalb eines vorgegebenen Fensters liegt. Der Status zur Interpolation, der aktuellen Position und dem Überwachungsfenster können über die Schnittstelle abgefragt werden.

Positionsregelung

In der Betriebsart Positionsregelung erhält der Positionsregler die Sollposition zyklisch von der Steuerung (Bild 6.15). Die Parameter der Positionsregelung wirken ebenfalls, wenn die Sollposition aus dem Interpolator im Antrieb stammt. Die Sollposition wird abhängig vom Bezugspunkt, der Skalierung und dem Referenzoffset berechnet.

Die Differenz zwischen dem aktuellen Ist- und dem Sollwert wird dem Positionsregler zugeführt. Das dynamische Verhalten des PI- Positionsregler wird durch die Parameter für den Verstärkungsfaktor und die Integrationszeitkonstante beeinflusst. Zusätzlich können Verstärkungsfaktoren für die Geschwindigkeits- und Beschleunigungsvorsteuerung eingestellt werden. Die Stellgröße dient als Sollwert für die unterlagerte Kaskade zur Geschwindigkeitsregelung. Sie wird vorher auf Grenzwerte überprüft. Bei den Grenzwerten handelt es sich um Systemgrenzen, die abhängig von der Hardware sind (maximale Motordrehzahl). Zusätzlich können Grenzwerte von der Applikation vorgegeben werden, die unter den Systemgrenzen liegen (maximale Drehzahl im Handbetrieb).

Für die Istwerterfassung kann ausgewählt werden, über welchen Geber sie erfolgen soll. Der Istwert selbst kann zusätzlich durch ein parametrierbares Fenster überwacht werden. Als Status dienen der Schleppabstand und das Ergebnis der Fensterauswertung.

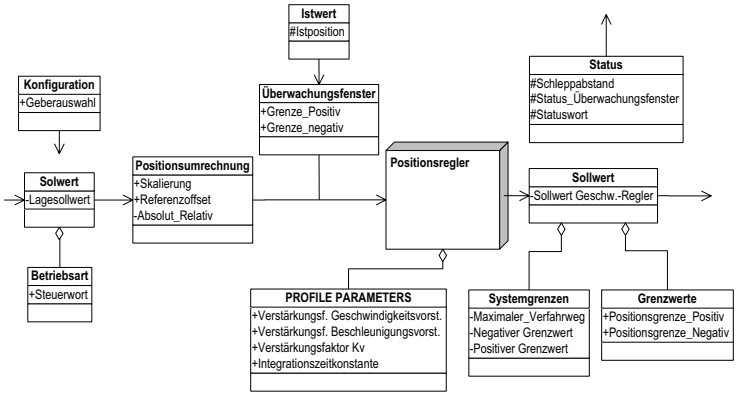


Bild 6.15: Betriebsart Positionsregelung und deren Struktur mit einheitlichen Parametern

Geschwindigkeitsregelung

Bei der Geschwindigkeitsregelung werden abhängig vom Steuerwort die Sollwerte aus dem Positionsregler oder zyklisch von der Steuerung übernommen. Die Sollwerte werden entsprechend der Skalierung und dem Offset umgerechnet.

Die Differenz zwischen Soll- und Istgeschwindigkeit wird an den Geschwindigkeitsregler übergeben. Der PID-Geschwindigkeitsregler stellt sein dynamisches Verhalten abhängig von den Parametern des Verstärkungsfaktors, der Integrationszeitkonstante und der Differentiationszeitkonstante ein. Die berechnete Stellgröße für den Momentenregler wird auf die Systemgrenzen hin geprüft. Die Applikation hat auch hier die Möglichkeit die Grenzen weiter herabzusetzen. Nach der Prüfung der parametrierbaren Grenzen wird die Stellgröße an den Momentenregler übergeben.

Für die Überwachung der Istgeschwindigkeit dienen auch hier Fensterfunktionen. Als Status kann der Zustand des Fensters und der Schlupf als Regelabweichung abgefragt werden.

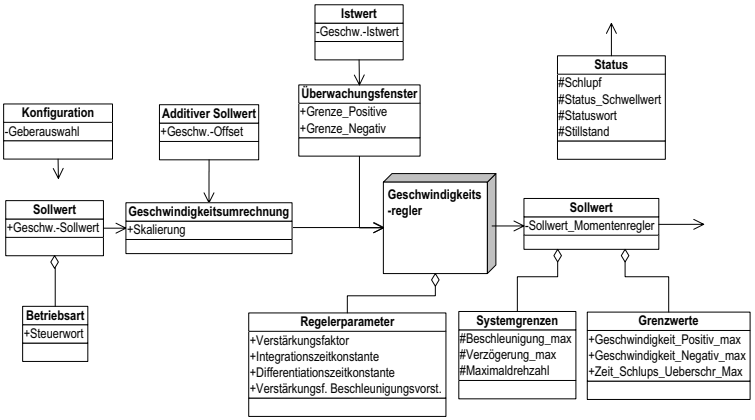


Bild 6.16: Betriebsart Geschwindigkeitsregelung und deren Struktur mit einheitlichen Parametern

Momentenregelung

Bei den Betriebsarten Momenten- und Stromregelung wird bei den Antriebsprofilen nicht unterschieden. Deshalb werden sie auch hier zusammen betrachtet.

Im Momentenregler werden die Sollwerte aus dem Geschwindigkeitsregler oder direkt aus der Steuerung übernommen. Die Betriebsart wird über das Steuerwort festgelegt. Die Abweichung des Istmoments vom Sollmoment wird im Momentenregler bzw. Stromregler jeweils mit dem Verstärkungsfaktor und der Integrationszeitkonstante zum Sollwert für den Umrichter berechnet. Davor wird der Stromwert auf die Systemgrenzen und parametrierten Grenzen überprüft. Als Status können die aktuellen Ströme und Spannungen am Motor sowie dessen Temperatur abgefragt werden.

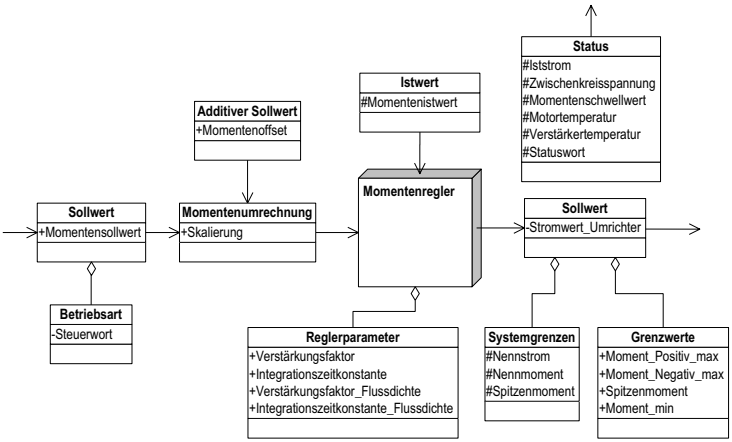


Bild 6.17: Betriebsart Momentenregelung und deren Struktur mit einheitlichen Parametern

Referenzieren

Bei der einheitlichen Betrachtung des Referenzierens wird zwischen antriebsgeführtem und steuerungsgeführtem Referenzieren unterschieden. Das antriebsgeführte Referenzieren lässt sich einheitlich, wie in Bild 6.18 dargestellt, beschreiben. Es werden für den Referenziervorgang Parameter für Richtung, Geschwindigkeit und Beschleunigung vorgegeben. Zusätzlich wird der Geber für die Positionserfassung ausgewählt und festgelegt, welche Flanke des Referenzschalters überwacht werden soll.

Beim Starten über ein Steuerbit fährt der Antrieb selbstständig durch die interne Interpolation bis zum Erreichen eines Referenzschalters oder Nullimpuls des Gebers. Der Antrieb berechnet selbst die Transformation zwischen Referenz- und Nullpunkt. Als Status kann der Zustand abgefragt oder der Abstand zwischen Referenz- und Nullpunkt über das Bussystem ausgelesen werden.

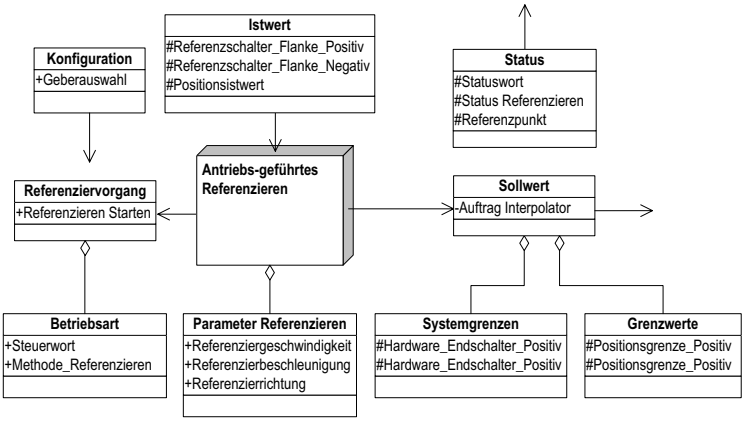


Bild 6.18: Betriebsart antriebsgeführtes Referenzieren und dessen Struktur mit einheitlichen Parametern

Beim steuerungsgeführten Referenzieren übernimmt die Steuerung selbst das Erzeugen der zyklischen Positionssollwerte (Bild 6.19). Sie überwacht ebenfalls die Referenzschalter selbst. Beim Erreichen der Referenzschalter berechnet die Steuerung die notwendige Transformation zum Referenzpunkt.

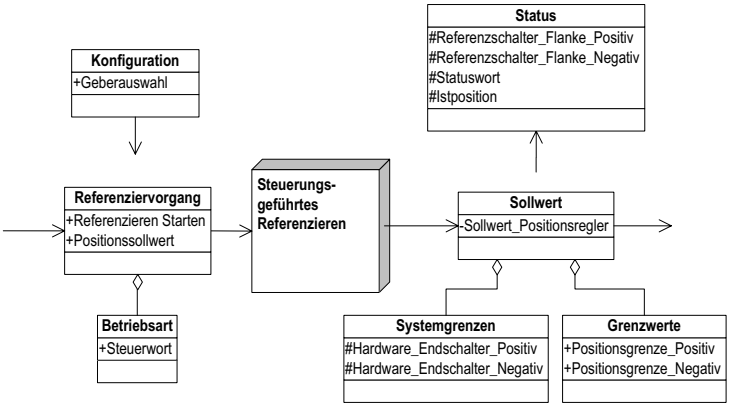


Bild 6.19: Betriebsart steuerungsgeführtes Referenzieren und deren Struktur mit einheitlichen Parametern

6.5 Zusammenfassung

In diesem Kapitel wurde, ausgehend vom vorherigen, detaillierten Vergleich eine funktional einheitliche Applikationsschnittstelle abgeleitet, die es erlaubt, die Funktionalitäten der Ethernet-basierten Bussysteme und Applikationsprofile einheitlich abzubilden. Dadurch lässt sich die Applikation von dem Kommunikationssystem trennen und es sind keine bussystem- und applikationsspezifischen Anteile in der Applikation notwendig.

Es wurden die für die Applikation relevanten Funktionalitäten der Kommunikation einheitlich abgebildet. Hierzu zählen der Hochlauf, die Konfiguration der logischen Verbindungen, sowie der zyklische und azyklische Datenaustausch.

Bei den Applikationsprofilen wurde ein einheitliches Gerätemodell abgeleitet, das eine Adressierung der angeschlossenen Teilnehmer und ihrer modularen Struktur ermöglicht. Zusätzlich wurde ein Parametermodell definiert, dass es erlaubt über dieselben Methoden einheitliche und busspezifische Parameter zu übertragen. Für den Datenaustausch der Werte wurde eine Struktur definiert, die sicherstellt, dass einheitlich auf die Vielzahl der unterschiedlichen Datentypen zugegriffen werden kann. Diese Struktur lässt sich unabhängig von der eingesetzten Hardware umsetzen.

Für die Antriebsprofile CiA 402, CIP Motion, Profidrive und FSP Drive wurde eine einheitliche Sichtweise auf die Antriebszustände und Betriebsarten definiert. Dabei wurde eine einheitliche Reglerstruktur hergeleitet, die sich auf die in den jeweiligen Antriebsprofilen spezifizierte Reglerstruktur abbilden lässt. Die einzelnen Betriebszustände wurden ausgehend vom Vergleich und dem hergeleiteten Entscheidungsbaum definiert und erläutert.

Diese funktional einheitliche Applikationsschnittstelle kann sowohl auf Seiten der Steuerung als auch auf Slaveseite umgesetzt werden. Es wurden darüber hinaus alle Anforderungen aus Kapitel 2 erfüllt. Die praxisnahe Umsetzung dieser Schnittstelle soll im nachfolgenden Kapitel nachgewiesen werden.

7 Realisierung

Die im vorhergehenden Kapitel hergeleitete funktional einheitliche Applikationsschnittstelle wird an einem Demonstrator getestet und bewertet. Die Realisierung der in dieser Arbeit hergeleiteten Ergebnisse wurde mit Unterstützung der Firma Homag Holzbearbeitungsmaschinen /96/ an einer ihrer Holzbearbeitungsmaschinen durchgeführt.

7.1 Beschreibung des Demonstrators

Bei dem Demonstrator handelt es sich um eine Holzbearbeitungsmaschine aus dem Bereich Durchlaufmaschinen der Firma Homag (Bild 7.1). Mit dieser Holzbearbeitungsmaschine werden Werkstückkanten vollautomatisch bearbeitet und Kantenfurnier angeleimt. Sie besteht aus mehreren einzelnen mechatronischen Modulen, die unterschiedliche Arbeitsschritte durchführen. Die einzelnen Module werden abhängig von Kundenanforderungen in die Maschine integriert und über das Kommunikationssystem mit der Steuerung verbunden.

Bild 7.1: Durchlaufmaschinen zur Holzbearbeitung der Firma Homag
(Quelle: Homag)

7.2 Beispielapplikation

Für die Umsetzung wird die Ansteuerung eines solchen mechatronischen Moduls aus dieser Holzbearbeitungsmaschine als Beispielapplikation gewählt. Bei dem betrachteten Modul FK 21 (Formfräsen/Kontur) handelt es sich um ein Modul, das zum Fräsen von Radien oder Fasen an den angeleimten Kanten der Werkstücke dient. Das Modul synchronisiert sich auf das durchlaufende Werkstück auf und bearbeitet die Kanten entsprechend (Bild 7.2). Dabei handelt es sich um eine typische Motionapplikation.

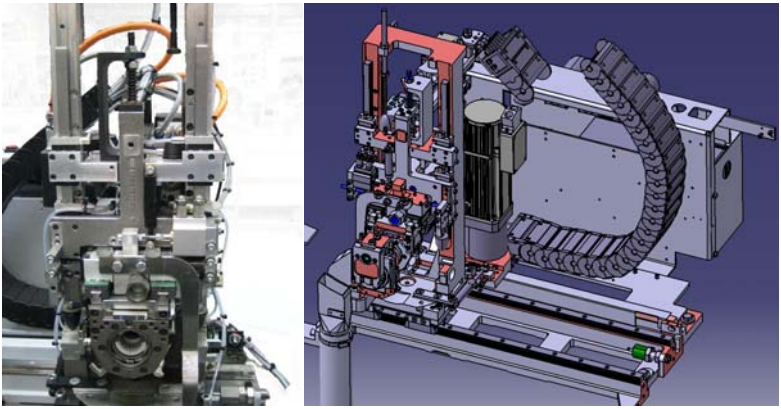


Bild 7.2: FK21 – Mechatronisches Modul als Beispielapplikation (Quelle: Homag)

7.2.1 Hardwarearchitektur

Das betrachtete mechatronische Modul FK21 besteht aus folgenden Geräten:

- Lokale Peripherie direkt an der Steuerung
- 3 digitale Servoantriebe mit Endschaltern
- 13 digitale Ein-und Ausgänge an 5 verschiedenen Klemmen hinter einem Buskoppler

Zur Ansteuerung der einzelnen Module kommt eine durch Homag selbst entwickelte PC-basierte CNC-Steuerung mit integrierter SPS zum Einsatz. Die Kommunikation zu den Geräten der Module erfolgt aktuell über CANopen und als Profil wird CiA 402 mit herstellerspezifischen Erweiterungen verwendet.

Das Modul kann ebenfalls mit Profidrive Antrieben, die über Profinet kommunizieren, ausgestattet werden. Dazu sind bisher tiefgreifende Anpassungen an der steuerungsseitigen Applikation in der CNC und SPS notwendig.

7.2.2 Systemarchitektur

Die Systemarchitektur der Steuerung mit dem integrierten LDI ist in Bild 7.3 dargestellt. Als Betriebssystem kommt ein Linux-Betriebssystem mit Echtzeiterweiterung von OSADL /97/ zum Einsatz.

Aus der Steuerungsebene greifen die CNC und SPS auf das LDI zu. Ein Wrapper übersetzt die funktional einheitlichen Zugriffe in busspezifische, die über das FDDI (Field Device Driver Interface) an den Bustreiber übergeben werden. Das FDDI entspricht dem BDDI aus Kapitel 6.1. Die darunter liegenden Ebenen kommunizieren über die in Linux vorhandenen Schnittstellen und reichen die Daten über die Hardware auf das Bussystem zur Übertragung an die angeschlossenen Teilnehmer weiter.

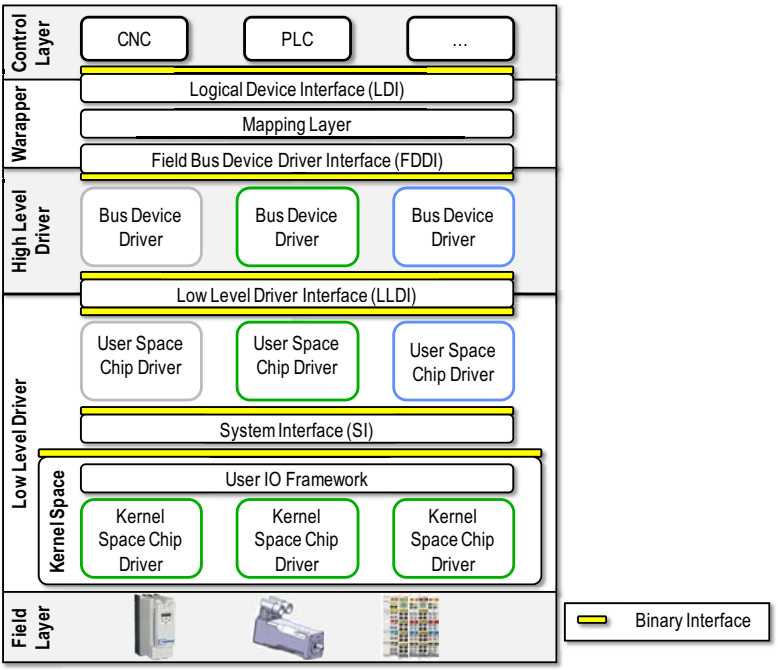


Bild 7.3: Einbindung des LDI und der Anpassungsschicht in die Steuerung der Firma Homag mit Echtzeit-Linux.

Die einzelnen Steuerungsmodule kommunizieren intern über einen gemeinsamen Speicherbereich, dem OCCI (Open Control Communication Interface). Das OCCI dient zum Informationsaustausch innerhalb der Steuerung und die Anbindung des Bussystemtreibers (Bild 7.4). Die funktional einheitliche Applikationsschnittstelle als LDI wird an das OCCI als einheitliche Schnittstelle zum Bussystemtreiber zwischengeschaltet.

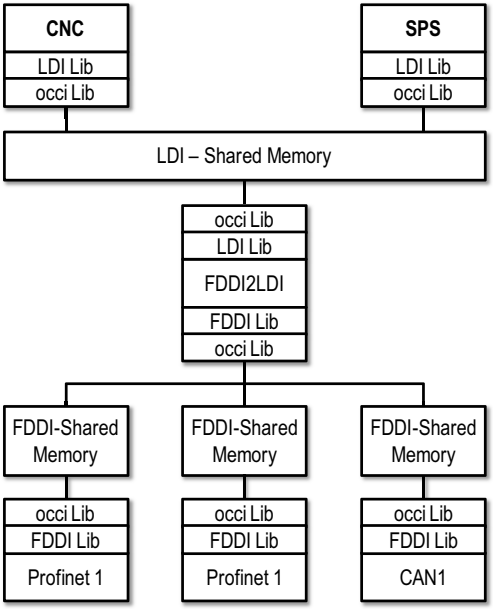


Bild 7.4: Steuerungsarchitektur mit OCCI (Quelle: Homag)

7.3 Ergebnisse

Die betrachtete Beispielapplikation verwendet folgende Funktionalitäten:

- Antriebsgeführtes Referenzieren
- Interpolation
- Positionsregelung
- Geschwindigkeitsregelung

Zur korrekten Funktion des FK21 werden insgesamt 58 azyklische und 8 zyklische Einzelfunktionen aus der Steuerung heraus genutzt. Von diesen genutzten Funktionen können 52 der azyklischen direkt über die funktional einheitliche Applikationsschnittstelle abgebildet werden. Bei den sechs nicht abbildbaren Funktionen handelt es sich um spezielle Erweiterungen des Antriebsverstärkers für die Firma Homag, die nicht im Antriebsprofil CIA 402 enthalten sind. Sie müssen über herstellerspezifische Parameter aus 6.4.1.1 über das LDI angesprochen werden. Die zyklischen Funktionen können alle über das LDI übertragen werden.

Der Wechsel von CANopen auf Profinet und dem Antriebsprofil CiA 402 auf Profidrive erfordert außer den Anpassungen für die sechs herstellerspezifischen Parameter keine weiteren Änderungen in den bestehenden Applikationen der CNC und SPS.

7.4 Zusammenfassung

In diesem Kapitel ist die Umsetzung der Ergebnisse dieser Arbeit an einer Beispielapplikation einer Holzbearbeitungsmaschine der Firma Homag dargestellt. Es konnte gezeigt werden, dass sich die in dieser Arbeit abgeleitete funktional einheitliche Applikationsschnittstelle erfolgreich umsetzen lässt.

Die für diese Applikation verwendeten Funktionen können bis auf 9% direkt über einheitliche Funktionen angesprochen werden. Die steuerungsseitige Applikation ist damit unabhängig vom verwendeten Kommunikationssystem und Antriebsprofil. Ein Wechsel auf ein anderes Bussystem erfordert somit kaum Anpassungen in der Applikation.

8 Zusammenfassung und Ausblick

Schwankungen bei den Stückzahlen, Produktlebenszyklen und Kosten erfordern Flexibilität und insbesondere Wandlungsfähigkeit in der Automatisierungstechnik. Um diese Wandlungsfähigkeit zu erreichen, entwickelt sich die Systemarchitektur aktueller und zukünftiger Maschinen, sowie Anlagen in Richtung dezentralisierter mechatronischer Module. Diese Entwicklungen führen zu einem erhöhten Kommunikationsaufwand innerhalb und zwischen solchen Modulen. Die Nutzung von Standard-Ethernet-Komponenten für die industrielle Kommunikation hat zu einer Erhöhung der Bandbreite, jedoch zu keiner Vereinheitlichung der industriellen Kommunikation geführt. Die einzelnen Kommunikationssysteme unterscheiden sich weiterhin in den Funktionalitäten zur Sicherstellung der Echtzeitfähigkeit und Synchronisation, sowie den Applikationsprofilen. Diese Unterschiede wirken sich direkt auf die Applikation aus. Das bedeutet, dass jede Applikation abhängig vom verwendeten Kommunikationssystem ist und bei einem Wechsel Anpassungen in der Applikation vorzunehmen sind.

Um Automatisierungssysteme und insbesondere Anwendungsprogramme zu schaffen, die unabhängig von der eingesetzten Kommunikationstechnik sind, beschäftigt sich diese Arbeit mit der Konzeption einer einheitlichen Applikationsschnittstelle für Ethernetbasierte Bussysteme. Diese Schnittstelle ist funktional einheitlich und verfügt über die benötigten Funktionalitäten verfügbarer Kommunikationssysteme und Applikationsprofile.

Keiner der existierenden wissenschaftlichen Ansätze und industriellen Umsetzungen definiert eine funktionale bussystemunabhängige Schnittstelle zwischen der Applikation und der Kommunikation. Die Defizite der einzelnen wissenschaftlichen Ansätze sind im Wesentlichen die fehlende Universalität bezüglich der Branche und der Applikation. Sie betrachten die Applikationsprofile nicht. Bisherige Vergleiche von Bussystemen beschäftigen sich hauptsächlich mit der Performance unterschiedlicher Systeme. Diese Vergleiche eignen sich somit nicht als Grundlage für eine Vereinheitlichung aus der eine Schnittstelle abgeleitet werden kann. Bei industriellen Lösungen existieren keine Lösungen, die ohne busspezifische Informationen in der Applikation auskommen. Sie vereinheitlichen lediglich teilweise die Hardware der OSI-Schicht 1. Es sind beim Wechsel auf ein anderes Bussystem jedoch globale Anpassungen notwendig. Eine strukturierte Sichtweise auf industrielle Bussysteme existiert ebenfalls nicht.

Die vorliegende Arbeit befasst sich deshalb mit der Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme anhand der Objektorientierten Analyse. Dazu wird zunächst eine Abstraktion der betrachteten Bussysteme und Applikationsprofile durchgeführt. Dabei wird analysiert, welche Funktionalitäten die einzelnen Systeme und Profile bieten. Sie werden ausgehend von geeigneten Quellen identifiziert und in einem nächsten Schritt generalisiert. Dazu wird eine gemeinsame Abstraktionsebene zwischen den jeweiligen Bussystemen und Applikationsprofilen hergeleitet. Innerhalb dieser gemeinsamen Abstraktionsebene wird ein struktureller Zusammenhang in Form einer Klassifikation der Funktionalitäten abgeleitet. Die Struktur wird ausgehend von der Topologie eines Netzwerkes betrachtet. Diese Abstraktion bietet die Grundlage für einen anschließenden Vergleich.

Ausgehend von dem strukturellen Zusammenhang werden die einzelnen Funktionalitäten anhand der Spezialisierung näher betrachtet. Die Funktionalitäten werden dazu in Funktionen unterteilt. Auf Basis der Spezifikationen für Ethernet-basierte Bussysteme werden die Funktionen detailliert beschrieben. Die Funktionen werden zwischen den unterschiedlichen Bussystemen verglichen und sowohl Gemeinsamkeiten als auch Unterschiede herausgearbeitet. Der Grad der Gemeinsamkeiten bzw. Unterschiede schwankt zwischen den einzelnen Funktionalitäten.

Auf Basis des vorhergehenden Vergleichs wird in einem weiteren Kapitel die funktional einheitliche Schnittstelle abgeleitet. Dazu werden die vergleichbaren Funktionen und Funktionalitäten einheitlich beschrieben und in die Schnittstelle übernommen. Es ist hierbei insbesondere zu entscheiden, welche Funktionen und Funktionalitäten einheitlich beschrieben werden. Zusätzlich zu den funktionalen Inhalten der Schnittstelle werden die Schnittstelle an sich und deren Konfiguration definiert. Dabei wird die Kommunikation so vereinheitlicht, dass sich alle Bussysteme zur Applikation hin gleich darstellen. Für Antriebsprofile wird eine einheitliche Parameterstruktur definiert, die es erlaubt, auf einheitliche und busspezifische Parameter über dieselbe Schnittstelle zuzugreifen. Die Datentypen der unterschiedlichen Bussysteme können alle auf einen 2 Byte Parameter mit 1 Byte Exponent und einem passenden Attribut zur Beschreibung der Einheiten abgebildet werden. Die Antriebsprofile werden nach den Kaskaden eines Antriebsreglers strukturiert und deren einheitliche Funktionen beschrieben.

Die Ergebnisse dieser Methode werden an einer modularen Holzbearbeitungsmaschine validiert. Es wird die Umsetzung der Ergebnisse dieser Arbeit an einer Beispielapplika-

tion einer Holzbearbeitungsmaschine der Firma Homag untersucht. Es kann gezeigt werden, dass sich die in dieser Arbeit abgeleitete funktional einheitliche Applikationsschnittstelle erfolgreich umsetzen lässt. Die für diese Applikation verwendeten Funktionen können bis auf herstellerspezifische Ausnahmen direkt über einheitliche Funktionen angesprochen werden. Die steuerungsseitige Applikation ist damit unabhängig vom verwendeten Kommunikationssystem und Antriebsprofil. Auch der Hersteller der verwendeten Komponenten spielt keine Rolle in der Steuerungsapplikation.

Die in dieser Arbeit hergeleitete Konzeption für eine funktional einheitliche Applikationsschnittstelle für Ethernet-basierte Bussysteme kann auf weitere Bussysteme mit zusätzlichen Anforderungen abgebildet werden. Hierbei sind beispielswiese Funkübertragungssysteme (Bluetooth, Zigbee,…) oder einfachste Sensorbusse (IO-Link, Lin, ...) denkbar. Dabei ist zu untersuchen, ob sich die zusätzlichen bzw. unterschiedlichen Funktionalitäten (Verbindungsaufbau, Topologie, …) aufgrund der Übertragungsphysik auf die Schnittstelle auswirken.

Darüber hinaus ist es vorstellbar, weitere Applikationsprofile (Sicherheit, Messtechnik, Pneumatik, Medizintechnik, …) anhand der in dieser Arbeit vorgestellten Methode zu vereinheitlichen. Ebenso stellt eine einheitliche Fehlerbehandlung und Diagnose von Bussystemen eine neue Herausforderung an weitere wissenschaftliche Arbeiten dar.

9 Literatur

/1/	Bullinger, H.-J.; Warnecke, H.J.; Westkämper, E.:	Neue Organisationsformen im Unternehmen. Springer Verlag, Berlin, 2008
/2/	Elger, J.; Haußer, C.:	Entwicklungen in der Automatisierungstechnik. In: Günther, W.; Hompel, M.: Internet der Dinge in der Intralogistik, Springer Verlag, Berlin, 2010
/3/	Wurst, K.-H.; Heisel, U.; Kircher, C.:	(Re)konfigurierbare Werkzeugmaschinen – notwendige Grundlage für eine flexible Produktion. In: wt Werkstattstechnik online 96 (2006), Nr. 5, S. 257-265
/4/	Nyhuis, P.; Gunther, R.; Abele, E.:	Wandlungsfähige Produktionssysteme: Heute die Industrie von morgen gestalten. PZH Produktionstechnisches Zentrum, 2008
/5/	Verl, A.; Kremer, M.; Kircher, C.; Sekler, P.:	Steuerungstechnik - Quo Vadis?. Fertigungstechnik für die Zukunft FTK, Stuttgart. 20./21. Sept. 2006, S. 77-98
/6/	Wörn, H.; Bringkschulte, U.:	Echtzeitsysteme Grundlagen, Funktionsweisen, Anwendungen. Springer-Verlag, Berlin, 2005
/7/	Pritschow, G.:	Automatisierung in der Produktion, Einführung in die Steuerungstechnik. Hanser Verlag, München, 2005

/8/	N.N.:	Kompass Steuerungstechnik. VDMA Arbeitskreis Steuerungstechnik, Elektrische Automation, Frankfurt, 2009
/9/	Töpfer, H.; Kriesel, W.:	Zum Generationswechsel bei Automatisierungssystemen. In: Regelungstechnische Praxis 24, 1982, S.336-341
/10/	Pfeifer, T.; Heiler, K.-U.:	Ziele und Anwendungen von Feldbussystemen. In: Automatisierungstechnische Praxis 29, 1987, S. 549-557
/11/	Wanser, K.:	Entwicklungen der Feldinstallation und ihrer Beurteilung. In: Automatisierungstechnische Praxis 27, 1985, S. 237-240
/12/	Gira, F.; Manninger, M.:	Feldbussysteme P-Net. Vorlesungsmanuskript, Institut für Computertechnik der Technischen Universität Wien, Wien, 1996
/13/	Kief, H.; Roschiwal, H.:	NC/CNC Handbuch 2007/2008. Hanser Verlag, München, 2007
/14/	Schmitz, S.; Wurst, K.-H.; Lechler, A.:	Integrierte mechatronische Module für rekonfigurierbare Bearbeitungssysteme. IFM2007, Winterthur, Schweiz, 12./13. September 2007
/15/	N.N.:	IEC 8802-3, Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, Beuth Verlag, Berlin, 2000

/16/	Sauter, T.; Wollschläger, M.:	Feldbussysteme – Historie, Eigenschaften und Entwicklungstrends. In: it - Information Technology 42, 2000, S. 7-16
/17/	Lüder, A.:	Industrielle Verkabelung im Umbruch: vom Feldbus zum Industrial Ethernet: Ethernet-basierte Automatisierungsprotokolle. Vogel Industrie Medien, Würzburg, 2006
/18/	Kaolan, G.:	Ethernet's winning ways. In: Sepctrum IEEE, Ausgabe 38, 2001, S. 113-115
/19/	Weck, M.; Brecher, C.:	Werkzeugmaschinen, Automatisierung von Maschinen und Anlagen. Springer Verlag, Berlin, 2006
/20/	Rüppel, W.; Selig, A.; Wetter, O.:	Sercos mit neuem Profil. In: Computer und Automation, Ausgabe 11, 2007
/21/	Rostan, M.; Stubbs, J.E.; Dzilno, D.:	Ethercat enabled advanced control architecture. In: IEEE Advanced Semiconductor Manufacturing Conference (ASMC), San Francisco USA, 11./13. July 2010
/22/	Homburg, D.:	Technik aus erster Hand – Feldbus und Ethernet in der industriellen Praxis. RBS, Stutensee, 2009
/23/	Koller, G.; Rauscher, Th.; Sauter, Th.:	Der Einfluss von Feldbussen auf verteilte Regelungen. In: e & i Elektrotechnik und Informationstechnik, Volume 118, Nr. 11, Springer Verlag, Wien, 2009. S. 556-553

/24/ Schriegel, S.; Jasperneite, J.: Investigation of Industrial Environmental Influences on Clock sources and their Effect on the Synchronization Accuracy of IEEE1588. In: International IEEE Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS07), Vienna, 2007

/25/ Verl, A.; Sawodny, O.; Hafla, A.; Dietmair, A.: Potenziale der modellbasierten Regelungstechnik in der industriellen Steuerungstechnik von Werkzeugmaschinen und Robotern. In: Fertigungstechnisches Kolloquium, Stuttgart, 10./11. Sept. 2008, S. 29-76

/26/ Neumann, P.: Communication in Industrial Automation. What is going on?. In: Control Engineering Practice 15, 2007, S. 1332-1347.

/27/ Oglodin, V.; Lechler, A.; Klemm, P.: Maschinenübergreifender Informationsaustausch. Methode zum Vergleich von Maschinen für den Transfer von Erfahrungswissen. In: wt Wekstattstechnik online 100 (2010), Nr. 3, S. 162-168.

/28/ N.N.: IEC 61508 (VDE 0803), Funktionale Sicherheit sicherheitsbezogener elektrischer/ elektronischer/ programmierbarer elektronischer Systeme Teil 1-7. VDE Verlag, Berlin, 2006.

/29/ Lechler, A.; Verl, A.: SERCOS safety Conformizer - Ein Werkzeug zum Konformitätstest von SERCOS safety Komponenten. In: Fortschritt-Berichte VDI Reihe 2 Nr. 663 - Fertigungstechnik, Zuverlässigkeit und Diagnose in der Produktion, VDI Verlag, Düsseldorf , 2007, S. 28-38

/30/	Schnell, G.; Wiedemann, B.:	Bussysteme in der Automatisierung und Prozesstechnik. GWV Fachverlage, Wiesbaden, 2006
/31/	Felser, M.; Sauter, T.:	Standardization of Industrial Ethernet - the Next Battlefield?. In: IEEE International Workshop on Factory Communication Systems, Wien, 22.-24 Sept. 2004, S. 413 - 421.
/32/	N.N.:	IMS research http://www.imsresearch.com (11.10.2010)
/33/	Lechler, A.; Verl, A.:	Einheitliche Kommunikationsschnittstelle für den funktionalen Zugriff auf ethernetbasierte Bussysteme. In: SPS/IPC/Drives, 24.-27. November 2009, Nürnberg, VDE Verlag, 2009, S. 41-49.
/34/	Lechler, A.; Verl, A.:	Functional Interface for Universal Access to Ethernet-based Field Bus Protocols. WMSCI, 13th World Multi-Conference on Systemics, Cybernetics and Informatics, Orlando, 2009
/35/	N.N.:	ISO/IEC 7498-1:1994-11, Informationstechnik - Kommunikation Offener Systeme - Basis-Referenzmodell. Beuth Verlag, Berlin, 1994
/36/	Stein, E.:	Rechnernetze und Internet. Hanser Verlag, München, 2008
/37/	Felser, M.:	Real-Time Ethernet – Industry Prospective. In: Proceedings of the IEEE, Volume 93, No.6, 2005

/38/	Dietrich, D.; Reiter, H.; Sauter, Th.; Schweinzer, H.-J.:	Die zweite Phase der Feldbusentwicklung. In: e&e Elektrotechnik und Informationstechnik, Volume 114, Nr. 5, Springer Verlag, Heidelberg, 1997, S 225-230
/39/	Felser, M.:	Real Time Ethernet for Automation Applications In: Embedded Systems Handbook - Networked Embedded Systems, ISA Corporation, San Francisco, 2009
/40/	N.N.:	IEC 61784-2, Industrielle Kommunikationsnetze - Profile - Teil 2: Zusätzliche Feldbusprofile für Echtzeitnetzwerke basierend auf ISO/IEC 8802-3. Beuth Verlag, Berlin, 2009
/41/	N.N.:	IEC 61158-2, Industrielle Kommunikationsnetze - Feldbusse - Teil 2, Dienstfestlegungen des Physical Layer. Beuth Verlag, Berlin, 2008
/42/	N.N.:	IEC 61158-3, Industrielle Kommunikationsnetze - Feldbusse - Teil 3, Dienstfestlegungen des Data Link Layer (Sicherungsschicht). Beuth Verlag, Berlin, 2008
/43/	N.N.:	IEC 61158-4, Industrielle Kommunikationsnetze - Feldbusse - Teil 4, Protokollspezifikation des Data Link Layer (Sicherungsschicht). Beuth Verlag, Berlin, 2008
/44/	N.N.:	IEC 61158-5, Industrielle Kommunikationsnetze - Feldbusse - Teil 5, Dienstfestlegungen des Application Layer (Anwendungsschicht). Beuth Verlag, Berlin, 2008

/45/	N.N.:	IEC 61158-6, Industrielle Kommunikationsnetze - Feldbusse - Teil 6, Protokollspezifikation des Application Layer (Anwendungsschicht). Beuth Verlag, Berlin, 2008
/46/	Felser, M.:	Vom Feldbus-Krieg zur Feldbus-Koexistenz. In: Bulletin des Schweizerischen Elektrotechnischen Vereins (SEV), Fehraltorf, Ausgabe 9, 2002, S. 24 -26
/47/	Staudt, S.:	Spezifikation und Konformitätstest zur Interoperabilität von automatisierten Produktionsmaschinen. Jost- Jetter Verlag, Heimsheim, 2006
/48/	Binder, A.; Wick, A.; Pollmeier, S.:	Vernetzte Antriebe : aktueller Stand und Trends. In: etz Elektrotechnische Zeitschrift, Ausgabe 7-8, 2003, Seite. 84-93
/49/	N.N.:	IEC 61800-7, Adjustable speed electrical power drive systems. Beuth Verlag, Berlin, 2007
/50/	N.N.:	EN 50325-4, Industrial communications subsystem based on ISO 11898 (CAN) for controller-device interfaces. Beuth Verlag, Berlin, 2003
/51/	N.N.:	ISO 11898, Road vehicles - Controller area network (CAN). Beuth Verlag, Berlin, 2003
/52/	Felser, M.:	IEC 61800-7: Die universelle Schnittstelle zur Antriebstechnik, ifm 2006 - Internationales Forum Mechatronik, Linz, 16./17. Oktober 2006

/53/ Felser, M.: Is a Generic Interface for Power Drive Systems possible?. 10th IEEE International Conference on Emerging Technologies and Factory Automation, Catania, 19.-22 Sept. 2005

/54/ Mahalik, N. P.: Fieldbus Technology. Springer Verlag, Berlin, 2003

/55/ Busse, R.: Feldbussysteme im Vergleich. Pflaum Verlag, München, 1996

/56/ Li, Y.: Bewertung der Echtzeitfähigkeit von Feldbussystemen. Fortschrittsbericht, VDI Reihe 10 Nr. 235, VDI-Verlag, Düsseldorf, 1993

/57/ Jasperneite, J.: Leistungsbewertung eines lokalen Netzwerkes mit Class-of-Service Unterstützung für die prozessnahe Echtzeitkommunikation. Shaker-Verlag, 2002

/58/ Wollert, J.: Industrielle Feldbusse. In: Gravatter H.-J.; Grünhaupt U.: Handbuch der Mess- und Automatisierungstechnik in der Produktion, Springer Verlag, Berlin, 2006

/59/ Schwager, J.: Ethernet erreicht das Feld. In: Elektronik, Ausgabe 11/2004, S. 48-54

/60/ Stöckel, T.: Kommunikationstechnische Integration der Prozessebene in Produktionssysteme durch Middleware-Frameworks. Fertigungstechnik - Erlangen 107, Meisenbach Verlag, Bamberg, 2000

/61/	Dietrich, C.:	Interface Layer und Profile für Feldbussysteme sowie deren formale Spezifikation. Magdeburg, Otto-von-Guericke-Universität, Fak. f. Elektrotechnik, 1994
/62/	Häberle, U.:	Einheitliche Anwenderschnittstelle für Feldbussysteme. Springer Verlag, Berlin, 1997
/63/	Rauscher, T.; Fischer, P.:	Ein branchenübergreifendes Klassenkonzept für Feldbusobjekte. In: it - Information Technology 4, 2000
/64/	Albrecht, H.; Meyer, D.:	Ein Metamodell für den operativen Betrieb automatisierungs- und prozessleittechnischer Komponenten. In: at – Automatisierungstechnik 50, R. Oldenbourg Verlag, München, 2002, S. 119-129
/65/	Schromm, H.:	Realisierung eines optimierten Feldbussystems und Modellierung mit Petrinetzen. Technische Universität Carolo-Wilhelmina zu Braunschweig, Fakultät f. Maschinenbau, 2003
/66/	N.N.:	Hilscher - Netx http://de.hilscher.com/netx.html (12.10.2010)
/67/	N.N.:	HMS Industrial Networks - Anybus, http://www.anybus.de (12.10.2010)
/68/	Jasperneite, J.; Imtiaz, J.; Schumacher, M.; Weber, K.:	A Proposal for a Generic Real-time Ethernet System, IEEE Transactions on Industrial Informatics(5), 2009, S. 75-85

/69/	Brecher, C.; Verl, A.; Lechler, A.; Servos, M.:	Open control systems: State of the Art. In:Production Engineering Volume 4, Numbers 2-3, 2010, S. 247-254
/70/	Koren, Y.; Jovane, F.; Pritschow, G.:	Open Architecture Control Systems – Summary of Global Activity. In: ITIA Series, Vol. 2, 1998
/71/	Pritschow, G.:	Offene, echtzeitfähige Steuerungsarchitektur. OCEAN: Open Controller Enabled by an Advanced Real-Time-Network. In: wt Werkstattstechnik online 93 (2003) 5, S. 374-378
/72/	Wenz, M.; Zimmermann, U.; Längle, T.; Wörn, H.:	Echtzeitfähige Komponentensoftware für die Entwicklung rekonfigurierbarer mechatronischer Systeme. In: Intelligente mechatronische Systeme, 2003, S93-105
/73/	N.N.:	FDT Interface Specification V1.2.1. FST Joint Interest Group, 2004
/74/	N.N.:	Device Type Manager (DTM) Style Guide V1.0. FST Joint Interest Group, 2005
/75/	Bender, K.; Großmann, D.; Danzer, B.:	Keeping it simple - Strategy for merging EDDL and FDT. In: Process Worldwide, Volume 10, 2007, S. 24-25
/76/	Simon, R.:	Field Device Tool-FDT Die universelle Feldgeräteintegration. Industrieverlag, Oldenbourg, 2003

/77/	Chappell, D.:	Understanding ActiveX and OLE: A Guide for Developers and Managers. Microsoft Press, 1996
/78/	N.N.:	PLCopen http://www.plcopen.org/ (19.10.2010)
/79/	DeMarco, T.:	Structured Analysis and System Specification. Prentice Hall, New Jersey, 1979
/80/	Coad, P.; Yourdon, E.:	Object-Oriented Analysis Second Edition. Yourdon Press Computing Series, Prentice Hall, New Jersey, 1991
/81/	Balzert, H.:	Lehrbuch der Objektmodellierung: Analyse und Entwurf. Spektrum, Akad. Verl. Heidelberg, Berlin, 1999
/82/	Booch, G.; u.a.:	Object-oriented analysis and design with applications, third edition. Addison-Wesley Professional, Amsterdam, 2007
/83/	Glinz, M.:	Informatik II. Vorlesungsmanuskript am Institut für Informatik, Zürich, Sommersemester 2007
/84/	N.N.:	ISO/IEC 19501, Information technology - Open Distributed Processing - Unified Modeling Language (UML) Version 1.4.2. Beuth Verlag, Berlin, 2005
/85/	N.N.:	Spezifikation Unified Modeling Language (UML) 2.3. Object Management Group (OMG), 2010

/86/	Klemm, P.:	Grundlagen der Prozessrechentechnik und der Softwaretechnik. Vorlesungsmanuskript am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen, Stuttgart, Wintersemester 2007
/87/	N.N.:	IAONA Handbook Industrial Ethernet. Industrial Automation Open Networking Alliance e.V.; Magdeburg, 2005
/88/	Born, M.; Holz, E.; Kath, O.:	Softwareentwicklung mit UML 2. Addison-Wesley, München,2004
/89/	N.N.:	Spezifikation Meta Object Facility (MOF) 1.3. Object Management Group (OMG), 2010
/90/	Lechler, A.; Selig, A.; Verl, A.:	Konzept für ein Testwerkzeug zur schnellen Entwicklung von konformen Ethernet-basierten Feldbusgeräten. In: SPS/IPC/Drives, Nürnberg, VDE, 27.-29. November 2007, S. 323-329
/91/	N.N.:	IEC/IEEE 61588, Precision clock synchronization protocol for networked measurement and control systems. Beuth Verlag, Berlin, 2004
/92/	N.N.:	DIN V 44302-2, Datenkommunikation; Fachwörter. Beuth Verlag, Berlin, 1994
/93/	Bronner, R.:	Planung und Entscheidung. 3. Auflage, Oldenbourg Verlag, München Wien 1999

/94/	Grünig, R.; Kühn, R.:	Entscheidungsverfahren für komplexe Probleme. 2. Auflage, Springer Verlag, Berlin, 2006
/95/	N.N.:	The International System of Units (SI). 8. Auflage, Organisation Intergouvernementale de la Convention du Mètre, 2006
/96/	N.N.:	Homag Holzbearbeitungssysteme, http://www.homag.com, (15.10.2010)
/97/	N.N.:	OSADL - Open Source Automation Development Lab, http://www.osadl.org/, Zugriff 15.10.2010

Lebenslauf

Persönliches	Name	Armin Lechler
	Geburtstag	04.11.1979
	Geburtsort	Bietigheim-Bissingen
	Eltern	Udo Lechler Gisela Lechler, geb. Schüle
	Ehefrau	Jasmin Lechler, geb. Braun
Schulbildung	1986 bis 1990	Grundschule Bietigheim-Bissingen
	1990 bis 1999	Gymnasium Bietigheim-Bissingen
Ersatzdienst	1999 bis 2000	Arbeiter-Samariter-Bund e.V. Ludwigsburg
Studium	2000 bis 2006	Studium der Technischen Kybernetik an der Universität Stuttgart
Berufstätigkeit	2006 bis 2009	Wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW)
	2009 bis 2011	Abteilungsleiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW)
	Seit 2009	Geschäftsführer der FISW Steuerungstechnik GmbH Stuttgart
	Seit 2011	Geschäftsführender Oberingenieur am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW)

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart und dem Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA

Herausgegeben bis Band 57 von Prof. Dr.-Ing. G. Stute †
ab Band 58 von Prof. Dr.-Ing. Dr. h. c.mult. Dr.-Ing. E.h G. Pritschow
ab Band 161 von Prof. Dr.-Ing. Dr. h.c. A. Verl

Erschienen bei Springer-Verlag:

1	Schmid, D.:	Numerische Bahnsteuerung, 1973.
2	Schwegler, H.:	Fräsbearbeitung gekrümmter Flächen, 1972.
3	Eisinger, J.:	Numerisch gesteuerte Mehrachsenfräsmaschinen, 1972.
4	Nann, R.:	Rechnersteuerung von Fertigungseinrichtungen, 1972.
5	Augsten, G.:	Zweiachsige Nachformeinrichtungen, 1972.
6	Karl, B.:	Die Automatisierung der Fertigungsvorbereitung durch NC-Programmierung, 1972.
7	Eitel, H.:	NC-Programmiersystem, 1973.
8	Knorr, E.:	Numerische Bahnsteuerung zur Erzeugung von Raumkurven auf rotationssymetrischen Körpern, 1973.
9	Bumiller, S.:	Viskohydraulischer Vorschubantrieb, 1974.
10	Maier, K.:	Grenzregelung an Werkzeugmaschinen, 1974.
11	Waelkens, J.:	NC-Programmierung, 1974.
12	Bauer, E.:	Rechnerdirektsteuerung von Fertigungseinrichtungen, 1975.
13	König, H.:	Entwurf und Strukturtheorie von Steuerungen für Fertigungseinrichtungen, 1976.
14	Damsohn, H.:	Fünfachsiges NC-Fräsen, 1976.
15	Jetter, H.:	Programmierbare Steuerungen, 1976.
16	Henning, H.:	Fünfachsiges NC-Fräsen gekrümmter Flächen, 1976.
17	Boelke, K.:	Analyse und Beurteilung von Lagesteuerungen für numerisch gesteuerte Werkzeugmaschinen, 1977.

18 Götz, F.-R.: Regelsystem mit Modellrückkopplung für variable Streckenverstärkung, 1977.

19 Tränkle, H.: Auswirkungen der Fehler in den Positionen der Maschinenachsen beim fünfachsigen Fräsen, 1977.

20 Stof, P.: Untersuchungen über die Reduzierung dynamischer Bahnabweichungen bei numerisch gesteuerten Werkzeugmaschinen, 1978.

21 Wilhelm, R.: Planung und Auslegung des Materialflusses flexibler Fertigungssysteme, 1978.

22 Kappen, N.: Entwicklung und Einsatz einer direkten digitalen Grenzregelung für eine Fräsmaschine mit CNC, 1979.

23 Klug, H. G.: Integration automatisierter technischer Betriebsbereiche, 1978.

24 Binder, D.: Interpolation in numerischen Bahnsteuerungen, 1979.

25 Klingler, O.: Steuerung spanender Werkzeugmaschinen mit Hilfe von Grenzregeleinrichtungen, 1979.

26 Schenke, L.: Auslegung einer technologisch-geometrischen Grenzregelung für die Fräsbearbeitung, 1979.

27 Wörn, H.: Numerische Steuersysteme – Aufbau und Schnittstellen eines Mehrprozessorsteuersystems, 1979.

28 Osofisan, P. B.: Verbesserung des Datenflusses beim fünfachsigen NC-Fräsen, 1979.

29 Berner, J.: Verknüpfung fertigungstechnischer NC-Programmiersysteme, 1979.

30 Böbel, K.-H.: Rechnerunterstützte Auslegung von Vorschubantrieben, 1979.

31 Dreher, W.: NC-gerechte Beschreibung von Werkstücken in fertigungstechnisch orientierten Programmiersystemen, 1980.

32 Schurr, R.: Rechnerunterstützte Projektsteuerung hydrostatischer Anlagen, 1981.

33 Sielaff, W.: Fünfachsiges NC-Umfangfräsen verwundener Regelflächen. Beitrag zur Technologie und Teileprogrammierung, 1981.

34 Hesselbach, J.: Digitale Lageregelung an numerisch gesteuerten Fertigungseinrichtungen, 1981.

35 Fischer, P.: Rechnerunterstützte Erstellung von Schaltplänen am Beispiel der automatisierten Hydraulikplanzeichnung, 1981.

36 Ackermann, U.: Rechnerunterstützte Auswahl elektrischer Antriebe für spanende Werkzeugmaschinen, 1981.

37 Döttling, W.: Flexible Fertigungssysteme – Steuerung und Überwachung des Fertigungsablaufs, 1981.

38 Firnau, J.: Flexible Fertigungssysteme – Entwicklung und Erprobung eines zentralen Steuersystems, 1982.

39 Herrscher, A.: Flexible Fertigungssysteme – Entwurf und Realisierung prozeßnaher Steuerungsfunktionen, 1982.

40 Spieth, U.: Numerische Steuersysteme – Hardwareaufbau und Ablaufsteuerung eines Mehrprozessorsteuersystems, 1982.

41 Schimmele, A.: Rechnerunterstützter Entwurf von Funktionssteuerungen für Fertigungseinrichtungen, 1982.

42 Sanzenbacher, M.: NC-gerechte Beschreibung von Werkstücken mit gekrümmten Flächen, 1982.

43 Walter, W.: Interaktive NC-Programmierung von Werkstücken mit gekrümmten Flächen, 1982.

44 Huan, J.: Bahnregelung zur Bahnerzeugung an numerisch gesteuerten Werkzeugmaschinen, 1982.

45 Erne, H.: Taktile Sensorführung für Handhabungseinrichtungen – Systematik und Auslegung der Steuerungen, 1982.

46 Plasch, D.: Numerische Steuersysteme – Standardisierte Softwareschnittstellen in Mehrprozessor-Steuersystemen, 1983.

47 Wang, Z. L.: NC-Programmierung – Maschinennaher Einsatz von fertigungstechnisch orientierten Programmiersystemen, 1983.

48 Schwager, J.: Diagnose steuerungsexterner Fehler an Fertigungseinrichtungen, 1983.

49 Klemm, P.: Strukturierung von flexiblen Bediensystemen für numerische Steuerungen, 1984.

50 Runge, W.: Simulation des dynamischen Verhaltens elektrohydraulischer Schaltungen – Einsatz von geräteorientierten, universellen Simulationsbausteinen, 1984.

51 Steinhilber, H.: Planung und Realisierung von Werkzeugversorgungssystemen für die NC-Bearbeitung, 1984.

52 Ohnheiser, R.: Integrierte Erstellung numerischer Steuerdaten für flexible Fertigungssysteme, 1984.

53 Keppeler, M.: Führungsgrößenerzeugung für numerisch bahngesteuerte Industrieroboter, 1984.

54 Kohler, P.: Automatisiertes Messen mit NC-Werkzeugmaschinen, 1985.

55 Rieger, K.-H.: Rechnerunterstützte Projektierung der Hardware und Software von Speicherprogrammierten Steuerungen, 1985.

56 Vogt, G.: Digitale Regelung von Asynchronmotoren für numerisch gesteuerte Fertigungseinrichtungen, 1985.

57 Chmielnicki, S.: Flexible Fertigungssysteme – Simulation der Prozesse als Hilfsmittel zur Planung und zum Test von Steuerprogrammen, 1985.

58 Renn, W.: Struktur und Aufbau prozeßnaher Steuergeräte zur Verkettung in flexiblen Fertigungssystemen, 1986.

59 Harig, K.: Quantisierung im Lageregelkreis numerisch gesteuerter Fertigungseinrichtungen, 1986.

60 Frank, H.: Programmier- und Überwachungsfunktionen für teileartbezogene NC-Werkzeugmaschinen, 1986.

61 Möller, H.: Integrierte Überwachungs- und Diagnose-Systeme für numerische Steuerungen, 1986.

62 Fink, H.: Einsatz speicherprogrammierbarer Steuerungen in der Fertigungstechnik, 1986.

63 Fleckenstein, J.: Zustandsgraphen für SPS – Grafikunterstützte Programmierung und steuerungsunabhängige Darstellung, 1987.

64 Wagner, E.: Steuerungen von Koordinatenmeßgeräten mit schaltenden und messenden Tastsystemen, 1987.

65 Grimm, W.: Diagnosesystem für steuerungsperiphere Fehler an Fertigungseinrichtungen, 1987.

66 Swoboda, W.: Digitale Lageregelung für Maschinen mit schwach gedämpften schwingungsfähigen Bewegungsachsen, 1987.

67 Gruhler, G.: Sensorgeführte Programmierung bahngesteuerter Industrieroboter, 1987.

68 Walker, B.: Konfigurierbarer Funktionsblock Geometriedatenverarbeitung für numerische Steuerungen, 1987.

69 Mayer, J.: Werkzeugorganisation für flexible Fertigungszellen und -systeme, 1988.

70 Lederer, R.: Programmierung von NC-Drehmaschinen mit mehreren Werkzeugschlitten, 1988.

71 Häberle, G.: NC-Musterprogrammierung für rechnerintegrierte Textilfertigung, 1988.

72 Pfeiffer, D.: Kompensation thermisch bedingter Bearbeitungsfehler durch prozeßnahe Qualitätsregelung, 1988.

73 Schmidt, W.: Grafikunterstütztes Simulationssystem für komplexe Bearbeitungsvorgänge in numerischen Steuerungen, 1988.

74 Egner, M.: Hochdynamische Lageregelung mit elektrohydraulischen Antrieben, 1988.

75 Schittenhelm, W.: Konfigurierbares Bedienungssystem für Steuerungen an Fertigungseinrichtungen, 1988.

76 Scheifele, D.: Grafisch dynamische Simulation des Bearbeitungsvorgangs für Doppelschlittendrehmaschinen, 1988.

77 Keuper, G.: Automatisierte Identifikation der Streckenparameter servohydraulischer Vorschubantriebe, 1989.

78 Kayser, K.-H.: Kollisionserkennung in numerischen Steuerungen mit der Distanzfeldmethode, 1989.

79 Viefhaus, R.: Fräsergeometriekorrektur in Numerischen Steuerungen für das fünfachsige Fräsen, 1989.

80 Zirbs, J.: Fertigungsgerechte Aufbereitung von Flächenverbänden bei der NC-Programmierung im Formenbau, 1989.

81 Ruoff, W.: Optische Sensorsysteme zur On-line-Führung von Industrierobotern, 1989.

82 Jantzer, M.: Bahnverhalten und Regelung fahrerloser Transportsysteme ohne Spurbindung, 1990.

83 Schumacher, H.: Einheitliche Programmierung von Automatisierungskomponenten roboterbestückter Bearbeitungs- und Montagezellen, 1991.

84 Schimonyi, J.: NC-Programmierung für das Werkzeugschleifen, 1991.

85 Wurst, K.-H.: Flexible Robotersysteme – Konzeption und Realisierung modularer Roboterkomponenten, 1991.

86 Hagl, R.: Erhöhung der Verfügbarkeit von Vorschubantrieben mit selbstanpassender Lageregelung, 1991.

87 Krebser, G.: Betriebssystem für NC mit einheitlichen Schnittstellen, 1992.

88 Lei, W.-T.: Flächenorientierte Steuerdatenaufbereitung für das fünfachsige Fräsen, 1992.

89 Diehl, G.: Steuerungsperipheres Diagnosesystem für Fertigungseinrichtungen auf Basis überwachungsgerechter Komponenten, 1992.

90 Nepustil, U.: Offene NC-Schnittstellen zur Korrektur von Fertigungsfehlern, 1992.

91 Bauder, M.: Konfigurierbare Robotersteuerung mit allgemeiner Transformation, 1992.

92 Philipp, W.: Regelung mechanisch steifer Direktantriebe für Werkzeugmaschinen, 1992.

93 Härdtner, G. M.: Wissensstrukturierung in Diagnoseexpertensystemen für Fertigungseinrichtungen, 1992.

94 Wiedmann, H.: Objektorientierte Wissensrepräsentation für die modellbasierte Diagnose an Fertigungseinrichtungen, 1993.

95 Rudloff, H.: Hochgenaue Konturerzeugung bei Bewegungsachsen mit einer dominanten mechanischen Resonanzstelle, 1993.

96 Brantner, K.: Adaptierbares Leitsteuerungssystem für flexible Produktionssysteme, 1993.

97 Kugler, W.: Kommunikationsmechanismen für offene Numerische Steuerungssysteme, 1994.

98 Schnurr, B.: Elektrodynamisches Antriebssystem zur Unrundbearbeitung, 1994.

99 Schneider, J.: Fehlerreaktion mit Speicherprogrammierbaren Steuerungen – ein Beitrag zur Fehlertoleranz, 1994.

100 Siewert, U.: Systematische Erstellung adaptierbarer Leitsteuerungssoftware am Beispiel der Durchsetzungsplanung, 1994.

101 Heger, G. F. J.: Maschinenferner Qualitätsregelkreis in flexiblen Fertigungssystemen, 1994.

102 Hofmeister, W.: Objektorientiert strukturiertes Programmiersystem für NC-Mehrschlittenmaschinen, 1994.

103 Horn, A.: Optische Sensorik zur Bahnführung von Industrierobotern mit hohen Bahngeschwindigkeiten, 1994.

104 Rentschler, U.: Fehlertolerantes Präzisionsfügen, 1995.

105 Junghans, G.: Modulares grafikunterstütztes Simulationssystem für Bearbeitungs- und Handhabungsvorgänge, 1995.

106 Heller, J.: Sensorgestützte Bewegungserzeugung leitlinienloser Transprtfahrzeuge, 1995.

107 Wieland, E.: Anwendungsorientierte Programmierung für die robotergestützte Montage, 1995.

108 Ketterer, G.: Automatisierte Inbetriebnahme elektromechanischer, elastisch gekoppelter Bewegungsachsen, 1995.

109 Reibetanz, Th.: Situationsorientierte Bearbeitungsmodellierung zur NC-Programmierung, 1995.

110	Frager, O.:	Durchgängige Programmierung von Fertigungszellen, 1996.
111	Ordenewitz, R.:	Betriebsweite Bereitstellung von Werkzeuginformationen, 1996.
112	Daniel, C.:	Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme, 1996.
113	Angerbauer, R.:	Anwenderorientierte Programmierung fahrerloser Transportsysteme, 1996.
114	Krauß, F.:	Splineverarbeitung in numerischen Steuerungen für das fünfachsige Fräsen, 1996.
115	Schittenhelm, K.-M.:	Einsatz vorgefilterter Führungsgrößen für Bewegungsachsen zur Bahnerzeugung, 1997.
116	Häberle, U.:	Einheitliche Anwenderschnittstelle für Feldbussysteme, 1997.
117	Strassacker, D.:	Testumgebung für die Implementierung und Inbetriebnahme eines adaptierbaren Leitsteuerungssystems, 1997.
118	Renz, B.:	Hochdynamische Strahllagekorrektursysteme zur Erhöhung der Bahngenauigkeit von CO_2-Laserbearbeitungsmaschinen, 1997.
119	Itterheim, C.:	Objektorientiertes Bearbeitungsmodell für Freiformflächen – Erstellung und maschinengebundene Modifikation –, 1997.
120	Müller, J.:	Objektorientierte Softwareentwicklung für offene numerische Steuerungen, 1997.
121	Glöckler, M.:	Verbesserung des Störverhaltens elektrohydraulischer lagegeregelter Zylinderantriebe, 1998.
122	Uhl, J.:	Entwurfssystematik für ein dezentral strukturiertes, objektorientiertes Fertigungsleitsystem, 1998.
123	Hammann, G.:	Modellierung des Abtragsverhaltens elastischer, robotergeführter Schleifwerkzeuge, 1998.
124	Scholich-Tessmann, W.:	Direktantriebe für Industrieroboter, 1998.
125	Wagner, R.:	Robotersysteme zum kraftgeführten Entgraten grobtolerierter Leichtmetallwerkstücke mit Fräswerkzeugen, 1998.
126	Anders, C.:	Adaptierbares Diagnosesystem bei Transferstraßen, 1998.
127	Fahrbach, C.:	Regelung hochdynamischer elektrischer Servo-Direktantriebe in Fertigungseinrichtungen, 1999.
128	Sperling, W.:	Modulare Systemplattformen für offene Steuerungssysteme, 1999.

Erschienen bei Jost-Jetter Verlag:

129 Kehl, G.: Gestaltung von Formgedächtnis-Aktorsystemen für sensorgeführte Inspektionsgeräte, 1999.

130 Brandl, Th.: Anlageninformationssystem - Informationsmodell und Erstellungssystematik, 1999.

131 Gronbach, H.: Simulationswerkzeug für die Gestaltung modularer CO_2-Laserbearbeitungsmaschinen, 1999.

132 Lutz, R.: Softwaretechnik für Maschinennahe Steuerungsfunktionen bei Fertigungseinrichtungen, 1999.

133 Tran, T. L.: Allgemeine Transformation für Maschinen mit Parallelkinematiken, 2000.

134 Bretschneider, J.: Reglerselbsteinstellung für digital geregelte, elektromechanische Antriebssysteme an Werkzeugmaschinen, 2000.

135 Schoenenberg, M.: Zuverlässiger Fertigungsprozess bei Transferstraßen durch präventive Maßnahmen, 2000.

136 Uhl, A.: Flexibles Telerobotersteuerungssystem auf der Basis offener numerischer Steuerungen, 2000.

137 Rui Li: Agentenbasierte NC-Planung für die Komplettbearbeitung auf Dreh-/Fräszentren, 2001.

138 Wildermuth, D.: Bahnvorbereitung in numerischen Steuerungen für Parallelkinematiken, 2001.

139 Handel, D.: Werkergerechte NC-Programmierung zur Komplett-Schleifbearbeitung von Bohrwerkzeugen, 2001.

140 Kosiedowski, U.: Adaptive Vorsteuerverfahren für elektromechanische Bewegungsachsen an Werkzeugmaschinen, 2001.

141 Haug, K.: Laser-Lichtschnittsensorik für die Automatisierung von Metall-Schutzgasschweißprozessen, 2002.

142 Hohenadel, J.: Einheitliches Steuerungssystem für NC und RC, 2002.

143 Wälde, K.: Sicherstellung der Softwarequalität von Anwendungsmodulen und Systemplattformen in offenen Steuerungssystemen, 2002.

144 Litto, M.: Störungsinformationssystem – Informationsmodell und Erstellungssystematik, 2002.

145 Schweiker, A.: Offene numerische Steuerungen für prozeßabhängige Bearbeitungen – vereinheitlichte Struktur, Funktionen und Schnittstellen – , 2003.

146 Rempp, B.: Regelungstechnische Untersuchung durchsatzgeregelter Produktionssysteme, 2003.

147 Eppler, C.: Kompensation fremderregter Schwingungen in Antriebssystemen mit Umlaufgetrieben, 2003.

148 Weiner, M.: Wiederverwendungsgerechte Entwicklungssystematik von Feinplanungssoftware für die flexible Fertigung, 2004.

149 McCormac, St.: Lageregelung hydraulischer Manipulatoren unter Einsatz eines Ferraris-Sensors, 2004.

150 Lehner, W.-D.: Regelung von Vorschubachsen unter Verwendung der Relativbeschleunigung, 2005.

151 Lewek, J.: Adaptierbares Informationssystem zur Erstellung baukastenbasierter Fertigungseinrichtungen, 2005.

152 Laible, U.: Aufbau numerischer Steuerungssysteme für sicherheitskritische Anwendungen, 2005.

153 Bürger, Th.: Durchgängige analytische Qualitätssicherung für numerische Steuerungssoftware, 2005.

154 Brinzer, B.: Produktionsregelung für die variantenreiche Serienfertigung, 2005.

155 Heusinger, S.: STEP-NC-basierter Korrekturkreis für die Schlichtbearbeitung von Freiformflächen, 2005.

156 Kirchberger, R.: Verbesserte Auswertung inkrementeller Messsysteme durch schnelle Signal-Vorverarbeitung, 2005.

157 Conrath, M.: Systematische Gestaltung von frequenzadaptierbaren Ultraschall-Werkzeugsystemen zum Einsatz in fertigungstechnischen Prozessen, 2005.

158 Wadehn, W.: Gestaltung von Antriebssystemen für formadaptive Strukturen, 2005.

159 Horber, H.: Fugendetektion bei Lichtbogenschweißprozessen mit robuster Signalverarbeitung für optische Sensoren, 2006.

160 Dreyer, J.: Situative Informationsbereitstellung an Fertigungseinrichtungen Informationsmodell und Erstellungssystematik, 2006.

161 Fritz, S.: Rekonstruktion von Prozesskräften aus Antriebssignalen von Werkzeugmaschinen, 2006.

162 Staudt, S.: Spezifikation und Konformitätstest zur Interoperabilität von automatisierten Produktionsmaschinen, 2006.

163 Reichle, R.: Einsatz von Internet-Werkzeugen und -Diensten in numerischen Steuerungssystemen für Werkzeugmaschinen, 2006.

164 Kaiser, L.: Systematik für das Qualitäts- und Projektmanagement bei der Abwicklung von Unikatprojekten, 2006.

165 Garber, Th.: Nutzung des redundanten Freiheitsgrades von sechsachsigen Parallelkinematik-Maschinen, 2007.

166 Altenburger, R.: Dynamische Eigenschaften und Regelung mechanisch verkoppelter Antriebssysteme, 2007.

167 Korajda, B.: Steuerungstechnische Verfahren zur echtzeitfähigen Kompensation der Fräserabdrängung, 2007.

168 Röck, S.: Echtzeitsimulation von Produktionsanlagen mit realen Steuerungssystemen, 2007.

169 Priese, J.: Verfahren zur durchgehenden dezentralen Planung in Werkstattstrukturen, 2007.

170 Boye, T.: Vorhersage der kinematischen Kalibriergüte von Parallelkinematiken, 2008.

171 Joannides, M.: Ein Beitrag zur volumenorientierten 3D-Objektrekonstruktion aus digitalen Daten, 2009.

172 Pruschek, P.: Verfahren zur anwendungsgerechten Parametrierung der Steuerung und Regelung von Vorschubachsen, 2009.

173 Fritsch, D.: Steuerung selbstorganisierender Multi-Roboter-Systeme für dynamische Sammelaufgaben am Beispiel der Bekämpfung maritimer Ölverschmutzungen, 2009.

174 Schmitz, S.: Industrielle Powerline-Kommunikation für Antriebseinheiten in Werkzeugmaschinen, 2010.

175 Bengel, M.: Workpiece-centered Approach to Reconfiguration in Manufacturing Engineering, 2010

176 Weimer, T.: Informationsmodell für die durchgängige Datennutzung in Fabrikplanung und -betrieb, 2010

177 Oglodin, V.: Maschinenübergreifender agentenbasierter Informationsaustausch für die Störungsbeseitigung, 2010

178 Kramer, C.: Offene Antriebsreglerplattform, 2011

179 Stotz, M.: Adaptive Segmentierung von Tiefenbildern für die 3-D-Objektlageerkennung auf Basis von kombinierten regelgeometrischen Elementen, 2011.

180 Selig, A.: Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten, 2011

181 Meyer, C.: Aufnahme und Nachbearbeitung von Bahnen bei der Programmierung durch Vormachen von Industrierobotern, 2011

182 Meier, M.: Verfahren zum emulationsgestützten MES-Engineering für die Photovoltaikindustrie, 2011

183 Walther, M.: Antriebsbasierte Zustandsdiagnose von Vorschubantrieben, 2011

184 Lechler, A.: Konzeption einer funktional einheitlichen Applikationsschnittstelle für Ethernet-basierte Bussysteme, 2011.