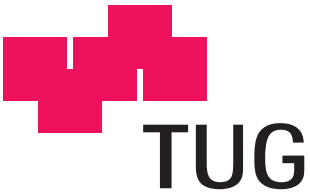
Masterarbeit

Echtzeitkommunikation im Bereich virtueller Motorenprüfstände

Harald Sporer, BSc.

Institut für Technische Informatik
Technische Universität Graz
Vorstand: O. Univ.-Prof. Dipl.-Ing. Dr. techn. Reinhold Weiß



Betreuer: Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Eugen Brenner

Graz, im Mai 2010

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am **11.05.2010** ..... Harald Sporer .....
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

11.05.2010 ..... Harald Sporer .....
date (signature)

Kurzfassung

Die gegenständliche Diplomarbeit setzt sich mit dem Aufbau eines Evaluierungs- und Testsystems auseinander, mit dessen Hilfe neue Hard- und Softwaremodule eines Komponentenorientierten Automatisierungssystems (AS) im Vorentwicklungsbereich getestet werden können. Der Vorteil dieses Systems ist, dass die Module nicht nur einzeln getestet werden können, sondern auch das Zusammenwirken der neuen AS-Teile validiert werden kann. Hauptaugenmerk nicht nur dieser Arbeit, sondern auch bei der Entwicklung des AS, liegt auf harter Echtzeit-Fähigkeit und minimaler Zykluszeit der eingesetzten Kommunikations-Technologie. Aus diesem Grund werden im ersten Teil der Arbeit die Grundlagen von Fast Ethernet und in weiterer Folge die Arbeitsweise der darauf aufbauenden Industrial Ethernets POWERLINK, PROFINET IRT sowie EtherCAT aufgezeigt. Diese drei Vertreter der Echtzeit-Ethernet-Lösungen wurden nicht zufällig ausgewählt. Diese repräsentieren die drei am meisten verwendeten Techniken für die Realisierung von Echtzeit-Ethernet. Dabei handelt es sich im Fall von POWERLINK um die Methode des Zeitschlitzverfahrens kombiniert mit Polling, bei PROFINET IRT um die exakte Planung der Übertragungszeiten und -wege realisiert mit Spezial-Switches und bei EtherCAT um die Technik des Summenrahmenverfahrens. Im Anschluss an die Beschreibung der drei Varianten folgt eine Leistungsbewertung und Gegenüberstellung der dargestellten Echtzeit-Ethernet-Varianten hinsichtlich ihrer maximal erreichbaren Performance.
Im zweiten Teil der Arbeit wird näher auf den Aufbau des Testsystems selbst, von der Hardware über die Software bis hin zu den verwendeten Simulationsmodellen, eingegangen und auch die Testszenarien selbst dargestellt. Die Funktionstüchtigkeit des aufgebauten Testsystems wurde anhand verschiedener Einsatzszenarien erfolgreich nachgewiesen.

Abstract

This master thesis presents the design and integration of an evaluation and test system in the field of automation systems that allows to test hardware and software modules in the predevelopment area. The main benefit of this system is the possibility to validate these new components not only separately but in interaction with each other.
One of the main tasks of the whole project is to reach hard real-time and minimal cycle times with the chosen communication technique. Therefore the basics of Fast Ethernet are presented at the first part of this thesis. Furthermore the techniques of the Industrial Ethernets POWERLINK, PROFINET IRT and EtherCAT are shown. These solutions represent the three main categories of enabling real-time behaviour at Ethernet. POWERLINK is using a combined method of time slicing and polling. In contrast PROFINET IRT is realized with special switches which needs exact planning of communication timing and communication paths. The last investigated technique EtherCAT is featuring the so called summation frame. At the end of this part of the thesis the presented Industrial Ethernets are compared to each other and evaluated regarding their maximal performance.
A closer look into the hardware and software of the test system is given in the second part of this thesis. The description of the used simulation models and some test scenarios is included as well. The full functionality of the system was proved by using different test procedures.

Danksagung

Die vorliegende Arbeit entstand im Studienjahr 2009/2010 am Institut für Technische Informatik der Technischen Universität Graz in Zusammenarbeit mit der AVL List GmbH.

Mein besonderer Dank gilt meinem Betreuer am Institut für Technische Informatik, Herrn Ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Eugen Brenner, der mir immer mit Rat und Tat zur Seite stand. Ohne seine Unterstützung wäre die Fertigstellung der Arbeit in dieser Form nicht möglich gewesen. Weiters möchte ich mich bei Herrn Prof. Dr.-Ing. Klaus Lebert bedanken, der für die Idee zu dieser Masterarbeit verantwortlich zeichnet und jederzeit für angeregte Diskussionen zur Verfügung stand. Ein herzlicher Dank auch an Frau Mag. Nicole Obermair für das Korrekturlesen dieser Masterarbeit.

Ein ganz besonderer Dank gilt meinen Eltern, die mich immer in jeglicher, erdenklicher Form unterstützt haben und immer hinter mir stehen.

Vielen Dank für all die Unterstützung!

Graz, im Mai 2010 Harald Sporer

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabenstellung

Innerhalb der Vorentwicklung für Prüfstandsautomatisierungssysteme werden Strategien untersucht, um modular und flexibel aufgebaute Soft- und Hardwarearchitekturen für zukünftige Prüfstandskonfigurationen zu entwickeln. Daraus abgeleitet wurden drei Teilprojekte definiert:

- Komponentenorientierte Soft- und Hardwarearchitekturen
- Zentrale Parametrierung und Visualisierung bei verteilten Strukturen
- Data Backbone und Kommunikationsprotokolle

Aus diesen Teilprojekten liegen derzeit erste Ergebnisse vor. Aufbauend auf dem existierenden Echtzeitbetriebssystem wurde eine Architektur entwickelt, die es erlaubt komponentenbasiert beliebige Signalflüsse aufzubauen und abzuarbeiten. Diese Signalkette kann über ein zentrales Parametrier- und Visualisierungswerkzeug konfiguriert werden. Diese neue Software-Architektur und die zugehörigen Werkzeuge, sollen im weiteren Verlauf dieser Arbeit als das zu testende Automatisierungssystem betrachtet werden.

Im Rahmen dieser Masterarbeit soll dafür ein Evaluierungs- und Testsystem aufgebaut werden, mit dem das Zusammenwirken der erzielten Ergebnisse validiert werden kann. Startpunkt ist ein zentraler Rechenknoten, auf dem zunächst ein rein simulierter Prüfstand in Betrieb genommen werden soll. Die vorhandene Simulationsbibliothek muss dazu ergänzt werden. Im zweiten Schritt soll das Gesamtmodell auf zwei Rechenknoten verteilt werden, die über ein Echtzeit-Ethernetprotokoll verbunden sind. Dadurch sollen einerseits umfangreiche Tests der neuen Software-Module des Automatisierungssystems ermöglicht werden und andererseits neue Hardware-Komponenten für die Kommunikation zwischen den Rechenknoten eingesetzt und gleichzeitig validiert werden.

Weiters soll, da das Thema Echtzeit-Kommunikation in allen Projekten rund um das neue Automatisierungssystems eine sehr zentrale Rolle spielt, das Thema Echtzeit-Ethernet im Detail betrachtet werden. Anhand von ausgewählten Vertretern des Industrial Ethernets sollen die verschiedenen Lösungswege beschrieben werden, um ein deterministisches Verhalten bei Ethernet zu erreichen. Die Vor- und Nachteile der verschiedenen Ansätze sollen dargestellt werden. Weiters soll eine Methodik eingeführt werden, um einen objektiven Vergleich der Leistungsfähigkeit der gezeigten Technologien zu ermöglichen.

1.2 Motivation

Die Anforderungen an neue Automatisierungssysteme sind vielfältig. Diese sollen unter anderem die Aufgaben Versuchsvorbereitung, Prüfstandssteuerung, Datenaufzeichnung und deren Auswertung erledigen. Weiters sollen sie skalierbar sein. Das bedeutet, es müssen von einfachen Komponententests bis hin zum komplexen, hochdynamischen Highend Antriebsstrangprüfstand viele Szenarien abgedeckt werden. Auch die Anforderungen bezüglich der angebundenen Messgeräte und I/Os werden immer höher und somit wird auch der Bedarf an schnelleren Kommunikationsmitteln mit höheren Durchsatzraten immer größer. Gleichzeitig sollen auch die Kommunikationsstrukturen vereinfacht werden. Eine große Anzahl von verschiedenen Übertragungs-Technologien ist unter allen Umständen zu vermeiden. Aufgrund dieser Forderungen gibt es große Bestrebungen modular aufgebaute, flexible Automatisierungssysteme mit "State of the Art"-Kommunikationstechniken zu entwickeln. Dies führte zu dem Wunsch einer genaueren Betrachtung von Echtzeit-Ethernet.

Diese neue Generation von Automatisierungssystemen benötigt neue Test- und Evaluierungssysteme. Aufgrund der Forderung nach modularen Komponenten, muss auch ein System geplant und aufgebaut werden, welches einen großen Bereich von Testaufgaben übernehmen kann. Nicht nur aus Qualitäts-Gründen muss eine Evaluierung, von sowohl neuen Hardware- als auch Software-Komponenten, in einem sehr frühen Entwicklungsstadium ermöglicht werden. Auch aus wirtschaftlicher Sicht (Zehnerregel der Fehlerkosten[1]) ist ein solches Testsystem unumgänglich.

1.3 Gliederung

In **Kapitel 2** findet eine generelle Beschreibung von Echtzeitsystemen statt. Es soll gezeigt werden, aus welchen Komponenten solche Systeme bestehen und wie deren Relationen zueinander sind. In diesem Rahmen wird auch ein kurzer Überblick über das verwendete Echtzeit-Betriebssystem gegeben. Weiters folgt ein detaillierter Einblick in Ethernet und dort im speziellen in den Standard Fast Ethernet, welcher die Grundlage für alle Industrial Ethernet-Lösungen darstellt.

In **Kapitel 3** wird eine Klassifizierung der Echtzeit-Ethernet Varianten, anhand ihrer Abweichung vom Fast Ethernet Standard und ihrer erreichbaren Performance, durchgeführt. Im Anschluss daran sollen die verschiedenen Ansätze in der Kategorie mit der höchsten Performance, anhand der Echtzeit-Ethernet-Varianten POWERLINK, PROFINET IRT und EtherCAT erläutert werden. Zum Abschluss dieses Kapitels wird eine Methode zur Leistungsbewertung eingeführt und auf die drei zuvor erwähnten Industrial Ethernet Vertreter angewandt.

Kapitel 4 beschäftigt sich mit dem Einsatz, der in den vorhergehenden Kapiteln dargestellten Echtzeit-Kommunikation. Zu diesem Zweck wird zu Beginn des Kapitels der Aufbau des Testsystems, aus Hard- und Softwaresicht, beschrieben und die Simulationsmodelle erläutert. Als Testszenarien werden stellvertretend für eine Vielzahl, an mit dem Testsystem realisierbaren Szenarien, die Folgenden beschrieben:

1. Software Tests

[1] Ein nicht entdeckter Fehler führt pro Schritt (Planung, Entwicklung, Fertigung, usw.) zu einer Verzehnfachung der Kosten, welche für seine Beseitigung notwendig ist [Rei10].

2. Tests am realen Prüfstand
3. Hardware-Komponenten Tests

In **Kapitel 5** folgen abschließende Bemerkungen, sowie ein Ausblick auf weitere Entwicklungen rund um die Themen dieser Arbeit.

Kapitel 2

Echtzeit- und Bussyteme

2.1 Echtzeitsysteme

2.1.1 Definition Echtzeitsysteme

Echtzeitsysteme stellen Rechner-Systeme dar, welche innerhalb von zeitlich vorgegebenen Grenzen mit einer vorhersagbaren Reaktion auf (unvorhergesehene) Ereignisse aus ihrer Umgebung reagieren müssen. Somit wird das korrekte Verhalten nicht nur durch die richtige Berechnung von Ergebnissen bestimmt, sondern auch durch den Zeitpunkt an dem das jeweilige Ergebnis vorliegt [Kop97]. Echtzeitsysteme können auf viele verschiedene Arten realisiert werden, egal ob es sich dabei um einfache Mikrocontroller oder um komplexe verteilte Rechner-Systeme handelt. Egal zu welchem Zeitpunkt und unter welchen Umständen, muss ein Echtzeitsystem die folgenden Eigenschaften besitzen:

- Rechtzeitigkeit: Die Tasks in einem Echtzeit-Computer-System müssen zu definierten Zeitpunkten mit ihren Berechnungen fertig sein.
- Gleichzeitigkeit: Es können unter Umständen mehrere Ereignisse aus der Umgebung des Echtzeitsystems gleichzeitig auftreten. Das System muss in solchen Fällen fähig sein, diese Ereignisse quasi-simultan abzuarbeiten.
- Berechenbarkeit / Vorhersagbarkeit: Ein Echtzeitsystem muss auf jedes Ereignis von außen mit einer vorhersagbaren Aktion reagieren.
- Zuverlässigkeit: Aus Sicht der Umgebung muss das Echtzeitsystem jederzeit stabil und zuverlässig arbeiten.

Für [Kop97] stellt sich ein Echtzeitsystem wie in Abbildung 2.1 gezeigt dar. Der "Operator" stellt den Menschen dar, der über das HMI, im einfachsten Fall Tastatur und Display, mit dem Echtzeit-Computer-System (RTCS) interagiert. Auf der anderen Seite steht das zu regelnde Objekt, mit dem das RTCS über definierte Schnittstellen, zum Beispiel Aktoren und Sensoren, kommuniziert. Der Mensch und das zu kontrollierende Objekt, werden auch als die Umgebung des Echtzeit-Computer-Systems bezeichnet. Ein solches Echtzeitsystem kann, wie zuvor bereits erwähnt, auch aus mehreren, verteilten Echtzeit-Computer-Systemen bestehen. Um diese miteinander zu verbinden und Daten auszutauschen, werden

Möglichkeiten zur Echtzeit-Kommunikation benötigt. Genau dieser Punkt stellt das zentrale Thema dieser Arbeit dar und wird in Kapitel 3 detailliert behandelt. Ein weiterer, wichtiger Punkt für ein Echtzeit-Computer-System stellt das Echtzeitbetriebssystem dar, welches im Folgenden überblicksmäßig erläutert wird.

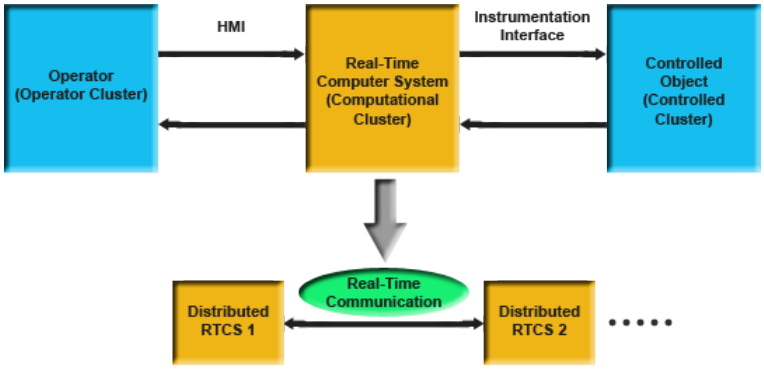


Abbildung 2.1: Echtzeitsysteme nach [Kop97]

2.1.2 Echtzeitbetriebssystem INtime

INtime[1] ist ein RTOS, welches von TenAsys speziell für die x86 Intel Architektur entwickelt wurde. Dabei handelt es sich nicht um ein eigenständiges Betriebssystem im herkömmlichen Sinn. INtime läuft je nach Rechnerarchitektur auf einer CPU gemeinsam mit Windows, beziehungsweise bei Multi-Core-Systemen auf einem eigenen CPU-Kern. Für die in dieser Arbeit beschriebenen Applikationen wären Systeme mit Single-Core nicht brauchbar, da die Kontext-Switches zu viel Zeit in Anspruch nehmen würden. Daher wird in den weiteren Ausführungen auf diese Variante nicht näher eingegangen. Bei Multi-Core Systemen wird die Anzahl der zu verwendenden Cores in den Windows Systemdateien beschränkt und somit ein Core für das RTOS reserviert. Durch diese Einschränkung des Nicht-Echtzeit Betriebssystems, können die Echtzeitprozesse und deren Threads ohne die zuvor angesprochenen Kontext-Switches arbeiten, was eine erhebliche Effizienzsteigerung mit sich bringt [Ten09].

Funktionsweise

Die Laufzeit-Umgebung der INtime-Software erweitert das bereits vorhandene Windows-Betriebssystem, sodass es möglich ist, Applikationen mit gewünschtem deterministischen Zeitverhalten laufen zu lassen. Es wird also ein RT Kernel zur Verfügung gestellt, welcher ein deterministisches Scheduling für die RT Threads bietet. Während der Initialisierungsphase des INtime Knotens wird für den RT Kernel und somit für die INtime Applikationen Speicher reserviert. Dieser Teil des Arbeitsspeichers wird gesperrt, sodass er nicht auf einen Massenspeicher ausgelagert werden kann. Weiters wird dieser auch vom "non-paged memory pool", welcher für Windows zur Verfügung steht, entfernt. Der RT Kernel verwendet für die Übersetzung der virtuellen Adressen den Paging-Mode des Prozessors (jedoch kein

[1] Informationen bezüglich in dieser Arbeit verwendeter eingetragener Marken und Markenzeichen siehe Anhang A.2

Demand-Paging). Jeder RT-Prozess lädt seinen Code, seine Daten beziehungsweise seinen Stack in einen eigenen virtuellen Adressraum, welcher zuvor vom RT Application Loader erzeugt wurde.

Kommunikation zwischen Real-Time Threads und Windows Threads

Eine INtime Applikation beinhaltet immer sowohl Real-time als auch Windows Prozesse. Die erst genannten enthalten jene Threads welche typischerweise zeitkritische Aufgaben abarbeiten (zum Beispiel I/O). Die Windows Prozesse hingegen enthalten nicht-zeitkritische Abläufe wie etwa die Benutzeroberfläche. Die Kommunikation und der Datenaustausch zwischen den RT Threads und den Windows Threads findet über eine Erweiterung der Win32 API, der so genannten NTX API, statt. Diese dient nicht nur zur Kommunikation zwischen den RT- und Windows Threads, sondern erkennt auch automatisch den Verbindungstyp und legt somit das Übertragungsprotokoll beziehungsweise das Übertragungsmedium fest. Dabei kann zwischen einer INtime Applikation auf einem Rechner und einer auf zwei oder mehreren Knoten verteilten unterschieden werden. Die in dieser Arbeit beschriebenen Applikationen gehören immer zur erst genannten Kategorie, weshalb auch auf eine genauere Betrachtung der verteilten Applikation verzichtet wird. Eine detaillierte Ausführung zum Thema verteilte Echtzeitapplikationen ist in [Gru08] zu finden. Es sei zur NTX API nur noch soviel gesagt, dass das Übertragungsmedium für eine Applikation auf einem Rechner der Arbeitsspeicher und für die verteilte Variante derzeit Ethernet beziehungsweise RS232 ist.

INtime auf Multi-Prozessor Systemen

Wie Eingangs bereits erwähnt, besteht die Möglichkeit Echtzeit-Anwendungen auf einem Single-Prozessor oder aber auf einem Multi-Prozessor System laufen zu lassen. Weiters kann bei Multi-Prozessor Systemen festgelegt werden, ob sich der INtime- und der Windows-Kernel eine (logische) CPU teilen und der Rest für Windows reserviert wird, oder aber eine (logische) CPU dem INtime-Kernel und der Rest dem Windows-Kernel zugewiesen wird. Mit letzterer Konfiguration lässt sich ein optimales zeitkritisches Verhalten des Systems erreichen. Der daraus folgenden Kontrollfluss ist in Abbildung 2.2 dargestellt. Der I/O APIC ist eine von zwei Komponenten, aus denen sich der Advanced Programmable Interrupt Controller (APIC) zusammensetzt. Dieser leitet die Interrupts, von zum Beispiel Geräten, als Interrupt-Message an den Local APIC, der zweiten Komponente des APIC, der betreffenden CPU weiter. Dieser wiederum leitet die Interrupts in der Reihenfolge ihrer Priorität an den CPU-Kern weiter. Somit werden vom I/O APIC Windows Interrupts nur an die Windows CPU und RT Interrupts nur an die INtime CPU weitergeleitet. Diese Konfiguration erlaubt es, dass Echtzeit-Prozesse niemals von Windows Interrupts oder Prozessen unterbrochen werden und zeitaufwendige Kontext-Switches gänzlich vermieden werden. Die Local APICs ermöglichen es auch, Interrupts zwischen den CPUs zu senden (Inter Processor Interrupt) [Ten08].

Schutz vor Blue screen

Der RT Kernel ermöglicht ein Weiterlaufen der RT Threads auch im Fehlerfall von Windows. Dies wird durch die Modifikation des Windows NT Hardware Abstraction Layers

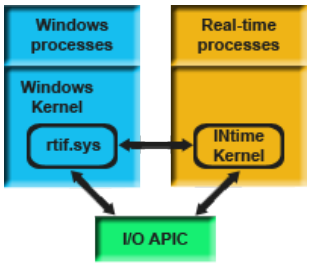


Abbildung 2.2: Kontrollfluss bei Multiprozessor-Konfiguration

(HAL) erreicht, wodurch der Windows-Fehler abgefangen und die Kontrolle komplett an den RT Kernel übergeben wird.

2.2 Bussysteme - Allgemeines

Unter einem Bussystem versteht man ganz allgemein den Austausch von verschiedenen Nachrichten zwischen mehreren Geräten über eine gemeinsame Leitung. Wird die Nachricht mittels Binärzahlen, welche in digitaler Form dargestellt werden, übermittelt spricht man von einem digitalen Bussystem. Die weiteren Ausführungen in dieser Arbeit beschränken sich auf diese Kategorie der Bussysteme, da es in der Automobilbranche keine analogen Bussysteme mit praktischer Bedeutung gibt [Bor08].

2.2.1 Einige Begriffe aus der Kommunikationstechnik

Für den weiteren Verlauf dieser Arbeit ist es notwendig einige Begriffe und Definitionen der Kommunikationstechnik kurz zu erläutern. So wird ein kurzer Überblick über die grundlegenden Begriffe wie Bandbreite, Bitrate, Übertragungsfrequenz und der Vollständigkeit halber auch der Baudrate gegeben. Die Kanalkapazität, der Jitter, der Unterschied zwischen Halb- und Vollduplex, die verbindungsorientierten versus verbindungslose Dienste, sowie Zugriffsverfahren werden im Folgendem ebenfalls kurz beschrieben.

Bandbreite

Die Bandbreite eines Übertragungssystems gibt den Frequenzbereich an, in welchem ohne wesentliche Beeinflussung - also mit geringer Dämpfung - übertragen werden kann. Zu erwähnen ist, dass Dämpfung und Frequenz proportional zusammenhängen, das heißt umso höher die verwendete Frequenz, desto höher ist auch die Dämpfung. Ein geeignetes Übertragungsmedium liegt vor, wenn die Bandbreite des Übertragungsmediums größer als jene des Datensignals ist.

Bitrate

Die Bitrate gibt die Anzahl der Bit an, welche in einer Zeiteinheit - üblicherweise in einer Sekunde - übertragen werden können (zum Beispiel 1 GBit/s). Hin und wieder wird die Bitrate auch als Datenrate bezeichnet.

Baudrate

Ursprünglich wurde die Baudrate im Bereich der Telegrafie - als Signalisierungsgeschwindigkeit der Telegrafen - verwendet. Heute ist sie definiert als Anzahl der pro Zeiteinheit übertragenen Symbole. Da diese Größe meist eher Verwirrung als Information liefert, wird im Weiteren darauf verzichtet.

Übertragungsfrequenz

Die Übertragungsfrequenz ergibt sich aus der Datenübertragung und wird oft fälschlicherweise mit der Datenrate gleichgesetzt. Die Datenrate drückt, wie zuvor erwähnt, die Anzahl der maximal über ein Übertragungsmedium zu übertragenden Bit pro Zeiteinheit aus. Wohingegen die Übertragungsfrequenz letztendlich vom verwendeten Codierungsverfahren abhängig ist. [Dan06] definiert die Begriffe Code und Codierungsverfahren mit:

> "Ein Code entsteht durch die Abbildung eines Quellenalphabets in ein Codealphabet, wobei das Codealphabet ganz bestimmte gewünschte Eigenschaften aufweist, die das Quellenalphabet nicht hat. Die Abbildung selbst muss umkehrbar eindeutig sein, damit die durch die Codierung veränderte Form der Informationsdarstellung wieder die ursprüngliche Gestalt annehmen kann. Die Abbildungsvorschriften sind die Codierungsverfahren."

Die gewünschte Eigenschaft, welche hier durch ein entsprechendes Codierungsverfahren erreicht werden soll, ist eine möglichst niedrige Übertragungsfrequenz. Wie schon zuvor erwähnt, wirkt sich die Höhe der Frequenz direkt auf die Dämpfung aus, was weiters auch die Reichweite beeinflusst. Daher ist das Ziel eine möglichst niedrige Übertragungsfrequenz, somit eine geringe Dämpfung und eine daraus resultierende möglichst große Reichweite.

Kanalkapazität

Die Kanalkapazität gibt an, wie viele Daten über einen frequenzbeschränkten, rauschbelasteten Übertragungskanal transportiert werden können. Die als Shannon'sche Informationstheorie bekannten Zusammenhänge besagen, dass die Kanalkapazität von der Bandbreite des Übertragungssystems, sowie vom Signal-Rauschverhältnis abhängt. Der von Claude Shannon beschriebene Zusammenhang ist in 2.1 dargestellt und kann in [Sha48] nachgelesen werden.

$$C = B \times \log_2 \left(1 + \frac{S}{N}\right) \tag{2.1}$$

Das Signal-Rauschverhältnis wird durch $\frac{S}{N}$ dargestellt, wobei S für die Leistung des Nutzsignals und N für die Leistung des Störsignals steht. Die Bandbreite B kommt als multiplikativer Faktor hinzu. Aus diesem Zusammenhang ist ersichtlich, dass umso besser (größer) das Signal-Rauschverhältnis ist, umso größer auch die Datenrate des Übertragungssystems ist [Rec08].

Jitter

Aus dem Englischen übersetzt bedeutet dieser Begriff "Schwankung" und wird in sehr vielen technischen Disziplinen für durchaus unterschiedliche Phänomene verwendet. Allgemein gültig ist die Aussage, dass Jitter die Abweichung eines Ereignisses bezeichnet. In den folgenden Ausführungen werden zum Beispiel in den Abschnitten 3.2 bis 3.4 Angaben des Herstellers beziehungsweise der zuständigen Organisationen über die maximale Größe des Jitters wiedergegeben. Diese zeitlichen Angaben beziehen sich rein auf die Schwankung der Laufzeit der Datenpakete und lassen eventuell weitere auftretende "Jitterquellen" außer Acht. Beispiele für diese weiteren Quellen wären etwa die Ungenauigkeit des Taktgebers oder auch direkt aus dem Echtzeitsystem entstehender Jitter.

Halbduplex versus Vollduplex

Von Halbduplex spricht man, wenn nur ein Kanal für das Senden und Empfangen von Daten vorliegt. Zu jedem Zeitpunkt kann also nur in eine Richtung kommuniziert werden. Dies war zu Beginn des Ethernet der Fall, als Koaxialkabel das Übertragungsmedium darstellten. Bei den heutigen Varianten (Twisted Pair und Lichtwellenleiter) stehen getrennte Übertragungskanäle für die Sende- und Empfangsdatenströme zur Verfügung. Dies bedeutet, es kann jederzeit in beide Richtungen kommuniziert werden (Vollduplex).

Verbindungorientierte versus verbindungslose Dienste

Man unterscheidet bei der Art der Datenübertragung zwischen verbindungsorientierten und verbindungslosen Diensten. Erst genannte würden zum Beispiel eine Punkt-zu-Punkt Verbindung in einem Netzwerk darstellen, welche für die Dauer des Datenaustausches geschaltet wird. Dadurch entsteht der große Nachteil, dass während des Bestehens dieser Verbindung andere Teilnehmer im Netz diese Leitung beziehungsweise den Kanal nicht nutzen können. Bei einem verbindunglosen Dienst wird das Medium von mehreren Teilnehmern gemeinsam genutzt. Die zu übertragenden Daten werden in Pakete zusammengefasst und Adressinformationen hinzugefügt. Ein Aufbau einer dedizierten Verbindung ist in diesem Fall nicht notwendig.

Zugriffsverfahren

Bei der Kommunikation zwischen verschiedenen Teilnehmern über einen gemeinsamen Bus, muss der gleichzeitige, schreibende Zugriff verhindert, beziehungsweise erkannt und korrekt behandelt werden. Diese Aufgabe wird von den Zugriffsverfahren erledigt und im Fall von Fast Ethernet als CSMA/CD bezeichnet. Genauere Ausführungen dazu sind unter 2.3.1 zu finden.

2.2.2 Kommunikationsreferenzmodell

Aufgrund der immer stärker wachsenden Bedeutung der Kommunikation zwischen Rechnern, wurde in den 1970er Jahren von der International Standards Organisation (ISO) eine Arbeitsgruppe geschaffen, welche sich mit der Standardisierung dieser Art von Kommunikation beschäftigte. Daraus entstand das 1983 verabschiedete ISO/OSI-Referenzmodell

(ISO-Norm 7498). Es ist seither die Grundlage eines jeden Standards im Bereich Rechnerkommunikation. Das Referenzmodell besteht aus sieben (abstrakten) Schichten und ist in Abbildung 2.3 dargestellt.

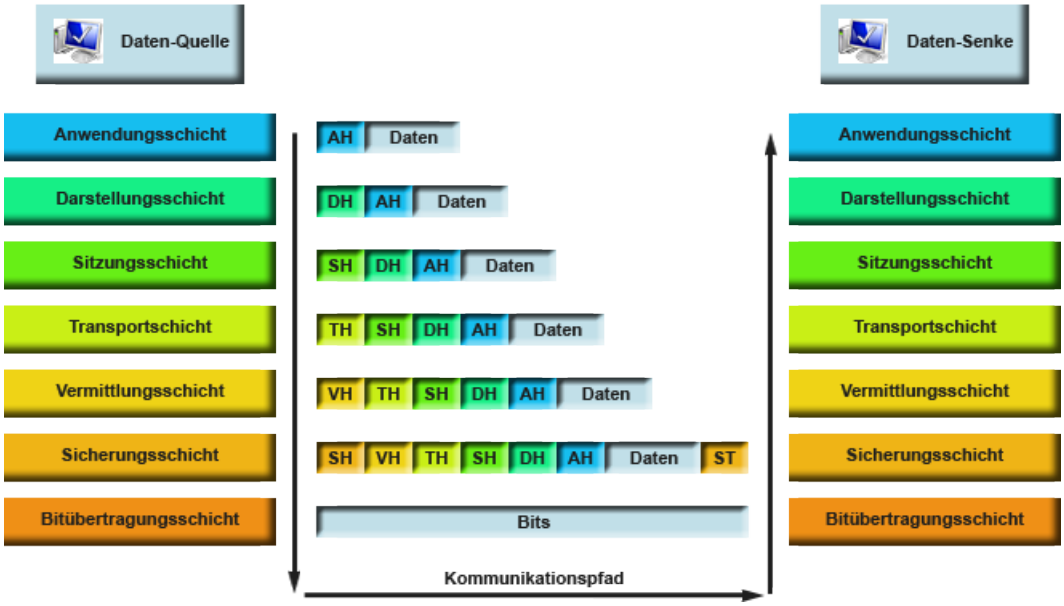


Abbildung 2.3: ISO/OSI-Referenzmodell

Jede dieser Schichten muss Aufgaben aus einem zugeordneten Teilbereich erfüllen und diese Funktion den benachbarten Schichten zur Verfügung stellen. Die Kommunikation zwischen den einzelnen Ebenen und somit der Zugriff auf die Funktionalität von benachbarten Schichten erfolgt über definierte Schnittstellen, den sogenannten Application Programming Interfaces (APIs). Diese exakt definierten Schnittstellen bilden die Grundlage für den problemlosen Austausch von verschiedenen Implementierungen der einzelnen Schichten.

Im Folgenden sollen die Aufgaben der einzelnen Schichten überblicksmäßig dargestellt werden. Eine genauere Spezifikation der Aufgaben und auch der Implementierung der einzelnen Schichten für Fast Ethernet ist im Abschnitt 2.3 zu finden.

Schicht 1 - Bitübertragungsschicht - Physical Layer

In der untersten Schicht des Referenzmodells werden die mechanischen und elektrischen Eigenschaften des Interfaces sowie das Übertragungsmedium definiert. Der in dieser Ebene bereitgestellte Dienst ist also für die physikalische Übertragung eines Bitstromes von einem Knoten zu einem Anderen zuständig. Zur Definition des Übertragungsmediums gehören auch die physikalischen Eigenschaften des Datensignals (Art und Pegel des Signals, Bitsynchronisation, Codierungsverfahren, u.ä.).

Schicht 2 - Sicherungsschicht - Data Link Layer

Der Data Link Layer ist für die Sicherstellung eines fehlerfreien Datenaustauschs zuständig. Dazu werden die zu übertragenden Bit und Byte zu Datenpaketen (Frames) zusammengefasst und üblicherweise durch einen Teil zur Fehlererkennung - zum Beispiel eine Prüfsumme - ergänzt. Ein weiterer wichtiger Teil dieser Ebene ist die Festlegung des Zugriffsverfahrens, welches eine kollisionsfreie Kommunikation der angeschlossenen Knoten sicherstellen soll.

Schicht 3 - Vermittlungsschicht - Network Layer

Die dritte Schicht kümmert sich um das Routing der Daten, also das Finden eines Weges zwischen zwei Knoten. Die zuvor beschriebenen beiden Schichten sind zwar in der Lage die Kommunikation zwischen zwei Knoten im selben Netzwerk aufzubauen, für einen Datentransfer zwischen Teilnehmern in verschiedenen Netzwerken wird jedoch die Funktionalität des Network Layers benötigt. Dazu werden den Daten entsprechende Ziel- und Quelladressen hinzugefügt.

Schicht 4 - Transportschicht - Transport Layer

Die Transportschicht sorgt dafür, dass die übermittelten Daten an die entsprechenden Applikationen beziehungsweise Dienste in den oberen Schichten weitergeleitet werden. Diese Funktionalität sorgt dafür, dass verschiedene Applikationen und Dienste auf den selben Transportmechanismus zugreifen können. Weiters werden in dieser Schicht die von oben kommenden Daten in kleinere Teile aufgeteilt, damit diese von der darunter liegenden Netzwerkschicht weiter verarbeitet werden können. Auf der Empfängerseite setzt der Transport Layer die fragmentierten Pakete wieder zusammen.

Schicht 5 - Sitzungsschicht - Session Layer

Der Session Layer wird im deutschen auch als Sitzungsschicht bezeichnet. Unter einer Sitzung versteht man die Verwendung der von der Transportschicht zur Verfügung gestellten logischen Kanäle. Diese Schicht bietet Dienste zum Auf- und Abbau von Sitzungen, wodurch ein oder mehrere Prozesse auf das Transportsystem zugreifen können. Weiters ist die Sitzungsschicht meist mit dem Betriebssystem verbunden und kann dadurch auch Prozesse synchronisieren. Dies ist teilweise erforderlich um einen korrekten Datenfluss zu gewährleisten.

Schicht 6 - Darstellungsschicht - Presentation Layer

Die sechste Schicht des Referenzmodells kümmert sich um die Darstellung der kommunizierten Daten. Dies beinhaltet Funktionen bezüglich des verwendeten Zeichensatzes, der Codierung, der Darstellung von Daten am Bildschirm oder Drucker, sowie auch sicherheitsrelevante Dienste, für zum Beispiel Ver- und Entschlüsselung der zu übertragenden Daten. Weiters gehört auch die Komprimierung von Daten zur Zeit- und Kostenersparnis bei der Übertragung zu den Aufgaben des Presentation Layer.

Schicht 7 - Anwendungsschicht - Application Layer

Die oberste Schicht stellt Funktionen zur Verfügung, mit denen der Benutzer auf das Kommunikationssystem zugreifen kann, wobei der Benutzer normalerweise nicht der Mensch sondern eine Applikation ist. Es werden anwendungsspezifische Protokolle zur Verfügung gestellt, wie etwa das File Transfer Protocol (FTP) oder auch das Hypertext Transfer Protocol (HTTP). Der Application Layer stellt außerdem eine Ortstransparenz der Daten her, das heißt die Applikationen dürfen nicht merken, dass logisch zusammengehörende Daten eventuell auf physikalisch verschiedenen Orten abgelegt sind [SW+06].

2.3 Fast Ethernet

Der Ethernet Standard spezifiziert die Aufgaben der Schichten Eins und Zwei des im Abschnitt 2.2.2 dargestellten Modells. Da das in dieser Arbeit verwendete Echtzeitprotokoll EtherCAT auf dem Ethernet Standard aufbaut, sollen dessen Grundlagen im Folgenden erläutert werden.

Das Haupteinsatzgebiet von Ethernet liegt im Bereich der lokalen Netzwerke (LANs). Durch seine ständige Weiterentwicklung konnte sich dieser Standard im Laufe der Zeit gegen seine direkten Konkurrenten, wie etwa Token Ring oder FDDI, weitestgehend durchsetzen und ist heute mit etwa 90 Prozent Anteil bei den installierten lokalen Netzwerken weit in Führung. Der ursprüngliche Ethernet Standard beinhaltete eine Übertragungsrate von 10 MBit/s; diese wurde über die Jahre auf 1 GBit/s weiterentwickelt und auch 10 GBit/s Lösungen sind bereits vorhanden. Weiters ist zu erwähnen, dass der Ethernet Standard mittlerweile in sehr vielen Variationen bezüglich der Übertragungsrate und des Übertragungsmediums zur Verfügung steht. Der zuvor angesprochene erste Standard - im Juni 1983 als Standard IEEE 802.3 verabschiedet - wurde zum Beispiel als 10BASE5 spezifiziert. Der erste Teil dieser Bezeichnung weist auf die 10 MBit/s Übertragungsrate, der zweite auf das Übertragungsverfahren - BASE entspricht dem Basisband, BROAD dem Breitband - und der dritte Teil auf die maximale Segmentlänge in 100 Meter beziehungsweise dem verwendeten Übertragungsmedium hin. In den darauf folgenden 25 Jahren entstanden mehr als 20 weitere Spezifikationen mit Übertragungsraten bis zu 10 GBit/s und verschiedensten Übertragungsmedien (Lichtwellenleiter, Twisted-Pair-Kabel, usw.) [Rec08]. Trotz dieser hohen Anzahl an Lösungen kann man sagen, dass die Daten im Wesentlichen immer auf die gleiche Art und Weise auf den Bus gebracht werden. Dies liegt einerseits am gleichbleibenden Zugriffsverfahren und andererseits am einheitlichen Aufbau des Ethernet Datenpakets. Das bedeutet, dass unterschiedliche Ansätze problemlos in einem Netzwerk betrieben werden können, ohne speziell Rücksicht auf eventuell unterschiedliche Datenraten beziehungsweise Übertragungsmedien nehmen zu müssen. Der Großteil der Lösungen im Bereich Ethernet und Echtzeitfähigkeit baut auf die Fast Ethernet Technologie auf. Aus diesem Grund wird auch im weiteren Verlauf dieser Arbeit der Ethernet Standard aus Sicht des 100BASE-TX betrachtet.

Mitte 1995 wurde der Standard IEEE 802.3u spezifiziert, welcher auch unter der Bezeichnung Fast Ethernet bekannt ist. Diesen Namen erhielt die 100BASE-X Lösung auf Grund der um den Faktor 10 gesteigerten Datenrate von 10 MBit/s auf 100 MBit/s. Unter IEEE 802.3u sind die Interfacevarianten 100BASE-TX (Twisted-Pair-Kabel der Kategorie 5), 100BASE-FX (Lichtwellenleiter) sowie die in der Praxis kaum eingesetzten 100BASE-

T2 und 100BASE-T4 (Twisted-Pair-Kabel der Kategorie 3) spezifiziert. Für die vorliegende Arbeit ist die Interfacevariante 100BASE-TX von Bedeutung, worauf sich auch die folgenden Ausführungen beziehen.

2.3.1 Fast Ethernet im OSI-Referenzmodell

In Abbildung 2.4 ist die Architektur von Fast Ethernet in Bezug auf das OSI-Referenzmodell dargestellt. Die einzelnen Unterschichten des Physical und Data Link Layer sollen im Folgenden kurz beschrieben werden.

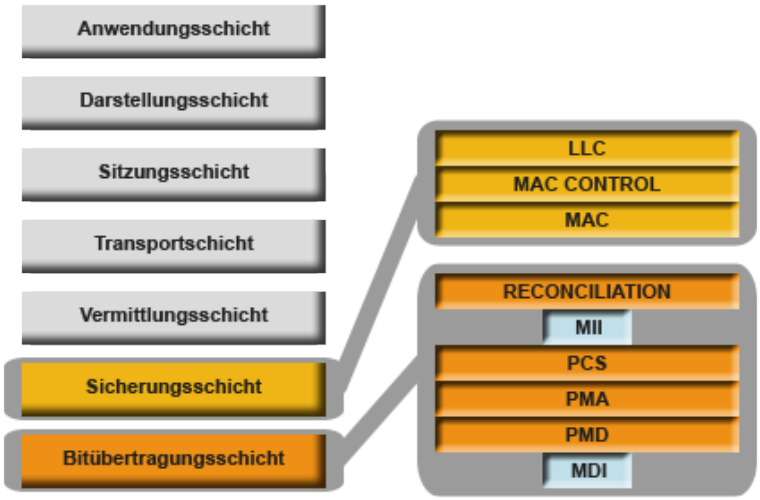


Abbildung 2.4: Physical und Data Link Layer bei Fast Ethernet

Logical Link Control (LLC)

Die oberste Teilschicht des Data Link Layer ermöglicht die gleichzeitige Verwendung von einer physikalischen Verbindung durch mehrere verschiedene, höhere Protokolle. Das LLC-Protokoll wurde im Standard IEEE 802.2 spezifiziert und stellt zusätzliche Ziel- und Quelladressierungsmöglichkeiten durch einen im Datenteil des Ethernet-Frames (siehe 2.3.2) integrierten Header zur Verfügung. Die LLC ist für die weiteren Betrachtungen nicht relevant und soll hier auch nicht näher ausgeführt werden.

MAC Control

Für diese optionale Teilschicht ist derzeit nur eine Funktionalität (Pause-Frame für Flow Control Mechanismus) spezifiziert. Die Ausführungen dazu folgen im Abschnitt 2.3.2.

Media Access Control (MAC)

Dieser Sublayer der MAC-Ebene beschreibt das Zugriffsverfahren mit Kollisionserkennung und zugehöriger Reaktion auf eine auftretenden Kollision (detaillierte Ausführung unter

2.3.1). Weiters ist hier spezifiziert, in welches Frame-Format die einzelnen Datenbyte gebracht werden müssen, um eine zielgerichtete Datenübertragung über das Netzwerk zu ermöglichen (siehe 2.3.2). Dazu gehört auch das Format der Ethernet-Adressen selbst.

Reconciliation

Aus dem Englischen übersetzt bedeutet "Reconciliation" soviel wie "Abgleich" oder auch "Abstimmung". Dies deutet auch auf die Funktionalität der obersten Teilschicht des Physical Layers hin: zusammen mit dem MII ist hier die Verbindung zwischen der MAC- und der PHY-Schicht spezifiziert, wobei der Reconciliation Sublayer dafür zuständig ist, die Signal der MAC-Schicht zu MII-Signalen zu transformieren (und vice versa).

Media Independent Interface (MII)

Diese Schnittstelle zwischen MAC und PHY besitzt ein Management-Interface zur Kontrolle und Konfiguration der unteren, physikalischen Teilschichten. Die Ausführung des Interfaces kann sowohl intern als auch extern erfolgen. Im erst genannten Fall verbindet das MII - über Netzwerkadapter - die MAC-Schicht direkt mit der physikalischen Schicht. Bei der externen Implementierung wird das Interface auf der Komponente über eine 40-polige MII-Buchse nach außen geführt und ermöglicht so den Anschluss eines Transceivers. Mit dieser Variante hat der Benutzer freie Wahl bezüglich des physikalischen Mediums [Rig05]. Die Signale des MII sind in der Tabelle 2.1 dargestellt.

Physical Coding Sublayer (PCS)

In dieser Teilschicht der physikalischen Ebene sind im Wesentlichen vier Aufgaben spezifiziert: Die je vier Bit breiten Sendedaten (Nibbles) werden zu fünf Bit breiten Codegruppen umgewandelt (siehe 4B/5B-Codierung unter 2.3.1) und die fünf Bit breiten Empfangsdaten werden wieder in die vier Bit breite Darstellung gebracht. Weiters erzeugt die PCS die in der Tabelle 2.1 angeführten Signale CRS und COL, welche die Grundlage des im MAC ausgeführten Zugriffsverfahrens darstellen. Ebenso erfolgt eine Konvertierung der Daten in eine für die serielle Übertragung günstige Darstellung, sowie die Signalanpassung zwischen PMA und MII.

Physical Medium Attachment (PMA)

Von unten gesehen ist die PMA die erste medienunabhängige Teilschicht des PHY und weist eine Kompatibilität zu allen bitorientierten, seriellen Medien auf. Es wird die Verfügbarkeit des PMD überprüft und auch der Takt aus dem NRZI-codierten Dateneingangsstrom (siehe 2.3.1) wieder hergestellt.

Physical Medium Dependent Sublayer (PMD)

Hier sind die möglichen Ausführungen des PMD spezifiziert (Glasfaser (FX) oder Kupfer (TX)).

Signal	*Bedeutung*
MDIO	Management Data I/O: Zusammen mit der MDC das MDIO Interface. Ermöglicht den Austausch von in den PHY-Registern gespeicherten Informationen (z.B. Link Status, Duplex Mode, usw.).
MDC	Management Data Clock
RXD<0> - RXD<3>	Receive Data Bit 0 - 3: Die Empfangsdaten werden in Gruppen von je vier Bit (Nibble) an die MAC weitergereicht.
RX_CLK	Receive Clock
RX_DV	Receive Data Valid: Zeigt den Empfang von Daten an.
RX_ER	Receive Error: In Kombination mit RX_DV und RXD können verschiedene Informationen über den Empfangsstatus kommuniziert werden.
TXD<0> - TXD<3>	Transmit Data Bit 0 - 3: Die Sendedaten werden in Gruppen von je vier Bit (Nibble) an die unteren Teilschichten weitergereicht.
TX_CLK	Transmit Clock
TX_EN	Transmit Enable: Zeigt das Aussenden von Daten an.
TX_ER	Transmit Error: In Kombination mit TX_EN können verschiedene Informationen über den Sendestatus kommuniziert werden.
COL	Collision: Beim Erkennen einer Kollision wird dieser Wert auf ”1” gesetzt (vom PCS generiert).
CRS	Carrier Sense: Durch eine ”1” wird ein bestehender Link angezeigt (vom PCS generiert).

Tabelle 2.1: MII-Signale

Medium Dependent Interface (MDI)

Im MDI sind die Medien-abhängigen Schnittstellen definiert - dies bezieht sich sowohl auf die physikalische Anpassung des Signals (zum Beispiel Anpassung der elektrischen Signalpegels), als auch auf die Spezifikation der Anbindung an das Übertragungsmedium (zum Beispiel Stecker-Aufbau).

Codierungsverfahren

Wie schon zuvor im Abschnitt 2.2.1 erwähnt, dient ein Codierungsverfahren immer dazu, dem Signal eine günstige Eigenschaft zu verleihen, welche es in seiner Ausgangsform nicht besitzt. Für die langsamere Ethernet-Variante mit einer Datenrate von 10 MBit/s wurde die Manchester-Codierung eingeführt. Diese garantiert ausreichend viele Pegelwechsel des Datensignals um daraus auch ein Taktsignal zur Synchronisierung gewinnen zu können. So stellt dieses Verfahren in der ersten Hälfte der Bitzeit den komplementären und in der zweiten Hälfte den tatsächlichen Wert des Bit dar. Die Bitzeit bei dieser Codierung beträgt 100 ns und die maximal auftretende Frequenz 10 MHz (bei 10 MBit/s Übertragungsrate). Geht man nun einen Schritt weiter zu einer Bitrate von 100 MBit/s ist leicht ersichtlich, dass die Manchester Codierung eine, für auf Kupfer basierende Übertragungsmedien, viel

zu hohe Übertragungsfrequenz von bis zu 100 MHz hervorrufen würde.

Fast Ethernet setzt auf die NRZI-Codierung (No Return to Zero/Inverted) und darauf aufbauend auf das 4B/5B-Codierungsverfahren. NRZI reduziert die Pegelwechsel - und somit auch die Übertragungsfrequenz - drastisch, da eine logische Eins als Pegelwechsel und eine logische Null als gleichbleibender Pegel dargestellt wird. Der Nachteil dabei entsteht bei einer langen Folge von logischen Nullen. Durch den nicht vorhandenen Pegelwechsel kann keine Information bezüglich des Taktsignals generiert werden. Als weiterer Nachteil ist auch der sogenannte "baseline wander" zu erwähnen. Dabei wirkt sich der gleichbleibende Pegel vergrößernd auf den Gleichanteil des Signals aus. Da der Ethernet-Link jedoch AC-gekoppelt ist, kann ein größer werdender Gleichanteil des Signal zu Übertragungsfehlern führen [CS09]. Um diesen Umstand zu verhindern, wurde die 4B/5B-Codierung eingeführt. Dabei werden jeweils vier Bit - sogenannte Nibbles - Benutzerdaten in fünf Bit lange Codegruppen umgewandelt. Diese garantieren genügend Taktinformation im Datenstrom. Weiters ermöglicht das zusätzliche Bit das Signalisieren eines bestimmten Status, beziehungsweise das Erkennen von Übertragungsproblemen, falls beim Empfänger eine Codegruppe eintrifft, welche eigentlich nicht gesendet werden darf (Violation-Symbole). Eine Auflistung der Codegruppen und ihrer jeweiligen Bedeutung kann zum Beispiel auf der Homepage der Universität Oldenburg gefunden werden [Old09]. Im, aus der Sicht der Übertragungsfrequenz, ungünstigsten Fall des Idle-Signals, welches aus fünf aufeinanderfolgenden Einsen besteht, tritt eine maximale Übertragungsfrequenz von 62.5 MHz auf.

MLT-3 und Scrambling

Zusätzlich zur erwähnten 4B/5B-Codierung wird noch MLT-3 ("Multilevel Threshold-3", oder oft auch als "Multilevel Transmission Encoding - 3 levels" bezeichnet) und Scrambling verwendet. MLT-3 stellt drei Werte (+1, 0, -1) zur Codierung zu Verfügung. Eine Folge von logischen Einsen wird alternierend als 0, +1, 0, -1, 0 usw. dargestellt. Bei einer logischen Null wird der letzte Pegel beibehalten. Diese Maßnahme reduziert die maximal auftretende Übertragungsfrequenz nochmals um 50 Prozent auf 31.25 MHz. Um die Länge von gleichbleibenden Pegeln nochmals zu reduzieren, wird als abschließende Maßnahme das Scrambling verwendet. Dabei werden die Bit in einer rekonstruierbaren Art und Weise verwürfelt, das heißt manche Einsen werden zu Nullen und vice versa. Die rekursive Funktion, der sich Scrambling bedient und Empfänger sowie Sender bekannt ist, ist in der Gleichung 2.2 dargestellt [Rec08].

$$X[n] = X[n-1] + X[n-9] \tag{2.2}$$

Diese Maßnahmen (NRZI, 4B/5B-, MLT-3-Codierung und Scrambling) stellen eine erhebliche Verbesserung im Vergleich zur einfachen Manchester-Codierung dar und ermöglichen den Betrieb von Fast Ethernet auf Twisted Pair als Übertragungsmedium.

Media Access Control - Protokolle

In der von unten gesehen zweiten Schicht, sind unter anderem die Protokolle für den Netzwerkzugriff spezifiziert. Dabei unterscheidet man, ob die Kommunikation über ein geteiltes Medium (Halb-Duplex), also einem gemeinsamen Übertragungskanal für den Sende- und Empfangsdatenstrom, oder über getrennte Kanäle (Voll-Duplex) stattfindet. Ein Beispiel

für den gemeinsamen Übertragungskanal wäre das Koaxialkabel, wohin gegen bei einem Twisted Pair (oder auch einem Lichtwellenleiter) getrennte Kanäle zur Verfügung stehen. Obwohl in den meisten Fällen, wie auch in den praktischen Ausführungen dieser Arbeit, bei Fast Ethernet ein Twisted Pair-Kabel der Kategorie 5 verwendet wird, ist aus Kompatibilitätsgründen auch in diesem Standard das Zugriffsverfahren für einen gemeinsamen Übertragungskanal spezifiziert.

Dieses Verfahren wird als "Carrier Sense Multiple Access/Collision Detection" (CSMA/CD) bezeichnet und dient dazu Kollisionen zu erkennen und notwendige Maßnahmen zu setzten. Dabei wird vom PCS, einem zuvor beschriebenen Sublayer der ersten Schicht, ein Carrier Sense-Flag zur Verfügung gestellt und über ein Interface beobachtet. Stellt dieses Flag den Wert "true" dar, übermittelt ein Teilnehmer im Netzwerk seine Daten und der Bus gilt als belegt. In diesem Fall werden die zu übermittelnden Daten in ein Ethernet-Frame verpackt und in eine Warteschlange gestellt. Sobald das Carrier Sense-Flag den Wert "false" darstellt, wartet der sendebereite Teilnehmer noch eine definierte Zeit, die sogenannte "inter-frame gap" (IFG), ab und beginnt anschließend mit dem Senden des Frames. Diese Maßnahme verhindert jedoch nicht, dass mehrere Teilnehmer den Bus als frei interpretieren und somit zeitgleich zu senden beginnen. Dieser Fall wird als "multiple access" bezeichnet und durch die "collision detection" erkannt, wobei die sendende Station während der Aussendung der Daten ständig überprüft, ob eine Kollision aufgetreten ist, oder das Frame erfolgreich übermittelt wurde. Im Falle einer Kollision wird der Sendevorgang unterbrochen und ein JAM-Signal als Broadcast ausgesendet um sicherzustellen, dass alle Teilnehmer die Kollision am Bus erkennen und somit den aktuellen Frame verwerfen. Anschließend wird von den sendewilligen Teilnehmern durch den Backoff-Algorithmus eine Zufallszahl generiert, welche als Wartezeit bis zum erneuten Sendeversuch dient. Auf eine genauere Beschreibung des Backoff-Algorithmus wird verzichtet, kann jedoch zum Beispiel in [Luk07] nachgelesen werden.

Des Weiteren ist noch zu erwähnen, dass eine Unterscheidung in zwei verschiedene Kollisionsarten zu machen ist. Eine "normale" Kollision tritt während der sogenannten "slot time" auf. Dies ist die benötigte Zeit zur Übertragung der ersten 64 Byte (Mindestlänge bei 100 MBit/s) eines Frames. Eine späte Kollision (engl. late collision) tritt zwischen Ende der slot time und dem Ende des Frames auf. Diese beiden Kollisionstypen werden unterschiedlich gehandhabt: Erstere führt zu einer Abhandlung wie zuvor beschrieben. Es wird die Übertragung gestoppt, ein JAM-Signal ausgesendet und nach einer zufälligen Zeit die Sendung wiederholt. Im zweiten Fall, der späten Kollision, wird die Übertragung beendet und der empfangene Frame verworfen. Der große Unterschied liegt darin, dass der Standard (bis inklusive 100 MBit/s) nicht zu einer Wiederholung der Sendung verpflichtet - hier kann das Verhalten zwischen verschiedenen Implementationen variieren.

Mit der Spezifizierung von Twisted-Pair-Kabeln als Übertragungsmedium wurde die Möglichkeit des Voll-Duplex Betriebes geschaffen. In einem Netzwerk mit Switches, oder einem Netzwerk mit lediglich zwei Teilnehmern in einer Kollisionsdomäne, kann jeder Knoten zeitgleich Daten senden und empfangen. Das bedeutet, dass zu sendende Daten in die Transmit-Warteschlange eingefügt werden und der jeweils erste Frame darin - nach dem Ende des IFG - gesendet wird, ohne Rücksicht auf die Belegung des Busses nehmen zu müssen. Somit ist das Zugriffsverfahren CSMA/CD überflüssig, was zu einer deutlich höheren Performance führt [SC05]. Jene Variante mit Voll-Duplex, Switches und lediglich einem Teilnehmer pro Port (Stern-Topologie) und daraus folgender kollisionsfreier Kom-

munikation, wird auch als "Switched Ethernet" bezeichnet.

Um einen Knoten vor einer Überlast durch einen zu großen Empfangsdatenstrom zu schützen, wurde der Flow Control Mechanismus in die Spezifikation (IEEE 802.3x) mit aufgenommen. Mittels dieser Technik hat der empfangende Teilnehmer die Möglichkeit dem sendenden Teilnehmer ein Frame mit der Aufforderung zu einer Sendepause zu übermitteln. Nähere Ausführungen dazu folgen im Abschnitt 2.3.2.

Auto-Negotiation Funktion (Normal Link Pulse vs. Fast Link Pulse)

Zur Überprüfung der bestehenden physikalischen Verbindung zwischen zwei Knoten (Link-Integritätstest) wird bei 10BASE-T der Normal Link Pulse (NLP) verwendet. Alle 16 ms $\pm$ 8 ms nach einem Frame beziehungsweise während einer Idle-Phase, wird ein Puls ausgesendet. Wird für die Dauer von 50 ms bis 150 ms kein NLP empfangen, gilt die Verbindung als unterbrochen. Diese Funktionalität wurde im Fast Ethernet Standard erweitert und der Fast Link Pulse (FLP) spezifiziert. Dabei werden statt den NLPs Link-Integrity-Test-Pulse Bursts ausgesendet. Das bedeutet, die 100 ns langen NLPs werden durch 33 Impulse ersetzt welche ebenfalls alle 16 ms $\pm$ 8 ms ausgesendet werden. Diese 33 Pulse unterteilen sich in 17 Rahmenpulse (erster, dritter, fünfter, ..., 33ster Puls) und 16 Datenpulse (zweiter, vierter, ..., 32ster Puls). Alle 125 µs wird eine Rahmenpuls ausgesendet. Wird zwischen zwei Rahmenpulsen ein weiterer Puls gesendet, repräsentiert dies eine logische "1". Fehlt der Datenpuls stellt dies eine logische "0" dar. So setzt sich mittels der 33 Pulse ein 16 Bit langes Datenwort zusammen welches für die maximale Bitrate sowie den Duplex-Modus des jeweiligen Ethernet-Teilnehmer steht. Mit dieser Kommunikation werden automatisch die Übertragungsparameter von miteinander verbundenen Ethernet-Geräten ausgehandelt [Rec08].

2.3.2 Frameformat

Bei Ethernet können vier verschiedene Frameformate unterschieden werden:

- Ethernet II
- Ethernet 802.3
- Ethernet 802.2
- Ethernet SNAP

Der grundlegende Aufbau ist bei allen Varianten sehr ähnlich. In Abbildung 2.5 ist jener eines Ethernet II-Frames dargestellt. Der einzige Unterschied zum Ethernet 802.3 Format liegt im zwei Byte breiten Typfeld, welches dort als Längenfeld verwendet wird. Dieser Typ gibt Auskunft über das in der dritten Schicht verwendete Protokoll. So würde zum Beispiel ein Eintrag von 0x800 auf die Verwendung des Internet Protocol (IP) hinweisen. Der Unterschied zwischen Ethernet II und den beiden letzteren, oben aufgelisteten Varianten liegt im Datenteil, welcher außer den Daten selbst, noch Informationen für höhere Schichten (Netzwerkschicht und darüber) zur Verfügung stellt. Da diese aber für die vorliegende Arbeit nicht relevant sind soll auch nicht näher darauf eingegangen werden.

Abbildung 2.5: Ethernet II-Frame

Start-of-Stream Delimiter (SSD)

Zwischen den Frames werden bei 100BASE-X Idle-Symbole (bestehend aus fünf aufeinander folgenden Einsen, siehe auch 2.3.1) über den Bus gesendet, um einerseits den Link zu testen und andererseits den Knoten die Synchronisierung zu ermöglichen. Um den anstehenden Start eines Frames deutlicher hervor zu heben, wird vor dem eigentlichen Frame eine Bitfolge, welche als Start-of-Stream Delimiter bezeichnet wird, ausgesendet. Diese wird durch eine Kombination der, in der 4B/5B-Codierung als J und K gekennzeichneten Symbole (je fünf Bit) dargestellt.

Präambel und Start-of-Frame Delimiter (SFD)

Die Präambel dient zur Synchronisation des Empfängers und soll diesen auf eine folgende Nachricht aufmerksam machen. Die Bit der ersten sieben Byte alternieren zwischen 1 und 0. Das achte Byte der Präambel wird Start-of-Frame Delimiter genannt und setzt sich ebenfalls aus einer wechselnden Folge von 1 und 0 zusammen. Nur die beiden letzten Bit enthalten eine logische 1, womit der Start des Frames bekannt gegeben wird.

Zieladresse

Die auf das SFD folgenden sechs Byte enthalten die Hardwareadresse des Empfängers. Durch diesen Eintrag werden die Daten entweder an einen (Unicast), mehrere (Multicast) oder alle (Broadcast) Knoten im lokalen Netz adressiert.

Quelladresse

Als nächstes Feld enthält der Frame die Hardwareadresse des Absenders. Diese wird vorwiegend von höheren Protokollen verwendet, welche mittels dieses Eintrags verifizieren, an welchen Teilnehmer eine eventuelle Antwort gesendet werden soll.

Länge

Das Längenfeld beinhaltet zwei Byte, welche die Anzahl der im Datenfeld zu übermittelnden Byte angibt. Wie bereits zuvor erwähnt, liegt hier der Unterschied zum Ethernet II Frameformat. Aus dem Inhalt dieser beiden Byte kann man somit auch auf das Frameformat schließen: Ist der enthaltene Wert größer als 1500, weißt dies auf Ethernet II und die Angabe des verwendeten höheren Protokolls hin.

Daten

Auf das Längenfeld folgen die eigentlichen Daten mit einer Länge zwischen 46 und 1500 Byte. Die Mindestframelänge von 64 Byte - 18 Byte Frameheader plus 46 Byte Daten - resultiert aus dem zur Kollisionserkennung verwendeten Zugriffsverfahren CSMA/CD. Für den Fall,

dass weniger als 46 Byte Daten in einem Frame verschickt werden sollen, wird dieses Feld mit Füllzeichen, sogenannten Pads mit der Wertigkeit 00, auf die Mindestlänge erweitert.

Cyclic Redundancy Check (CRC)

Das Frame Check Sequence Feld (FCS) stellt den letzten Teil des Ethernet Frames dar und dient der Fehlerüberprüfung. Dabei wird anhand der Daten aus Zieladresse-, Quelladresse-, Länge- und Daten-Feld durch den Cyclic Redundancy Check-Algorithmus eine vier Byte lange Prüfsequenz gebildet. Der Sender fügt diese Sequenz in das FCS Feld ein. Der Empfänger berechnet diese erneut und vergleicht sie mit jener aus dem FCS Feld des empfangenen Ethernet Frames. Sollte das berechnete vom empfangenen Ergebnis abweichen, wird der Frame als fehlerhaft interpretiert und verworfen. Die MAC-Schicht selbst führt keine Maßnahmen zur Fehlerbehebung durch, womit es die Aufgabe der übergeordneten Schichten ist, das verworfene Frame gegebenenfalls neu anzufordern. Zusätzlich zum fehlgeschlagenen CRC gibt es zwei weitere Fälle, in denen die empfangene Nachricht verworfen wird:

- die Anzahl der Byte im Datenfeld entspricht nicht jener der im Längenfeld deklarierten und
- die gesamte Länge des Frames ist kein Vielfaches eines Byte.

End-of-Stream Delimiter (ESD)

Als Gegenstück zum SSD folgt nach dem eigentlichen Ethernet Frame bei 100BASE-X der End-of-Stream Delimiter. Dieser wird durch eine Kombination der in der 4B/5B-Codierung als T und R gekennzeichneten Symbole (je fünf Bit) dargestellt. Durch den SSD und ESD werden die Grenzen des Frames besser hervorgehoben.

Flow Control

Zum Abschluss dieses Abschnitts soll das spezielle Frame des zuvor bereits erwähnten Flow Control Mechanismus dargestellt werden (siehe Abbildung 2.6). Diese Funktion wird grundsätzlich nur bei Voll-Duplex Peer-to-Peer Verbindungen eingesetzt. Als Zieladresse wird die Multicast-Adresse 01-80-C2-00-00-01 verwendet, welche von Switches und Routern nicht weitergeleitet wird und somit sicherstellt, dass nur der gegenüberliegende Teilnehmer das Pause-Frame erhält.

Abbildung 2.6: Ethernet Frame für Sendepause-Request

Der spezielle Typ wird mit dem Eintrag 0x8808 im Längenfeld und 0x0001 im MAC-Control-Opcode Feld bekannt gegeben. Zusammen mit dem darauf folgenden ebenfalls zwei Byte langen Parameter-Feld bildet das MAC-Control-Opcode Feld den MAC-Control-Frame. Das Pause-Frame ist derzeit das einzig spezifizierte MAC-Control-Frame. Das

Parameter-Feld enthält einen Wert zwischen 0 und 65535, welcher die Dauer der gewünschten Pause als ein Vielfaches der Slot time repräsentiert. Erhält der sendende Teilnehmer ein solches Frame, pausiert er für die gewünschte Dauer und setzt anschließend den Sendevorgang fort. Sollte der empfangende Teilnehmer - vor Ablauf der beantragten Pause - bereit sein neue Frames zu empfangen, besteht die Möglichkeit ein Pause-Frame mit dem Wert Null im Parameter-Feld zu versenden. Daraufhin wird die Sendepause abgebrochen und die Kommunikation wieder gestartet [Rec08].

Kapitel 3

Echtzeit - Ethernet

Aus den vorhergehenden Ausführungen kann man leicht schließen, dass bei herkömmlichen Ethernet-Lösungen kein deterministisches Zeitverhalten zu erwarten ist. Daher wurden im Laufe der Zeit verschiedenste Ansätze entwickelt um eben ein solches Verhalten zu erreichen. Der große Überbegriff hier lautet Industrial Ethernet. Darunter versteht man die Nutzung des mehr oder weniger abgewandelten Ethernet Standards zur Einbindung von Geräten im industriellen Umfeld - oder vereinfacht gesagt: die Verwendung von Ethernet als Feldbus. Eine umfangreiche Auflistung der Lösungen in diesem Bereich wurde von Prof. Dr.-Ing. Jürgen Schwager erstellt und ist auf der Homepage des Labors für Prozessdatenverarbeitung der Hochschule Reutlingen zu finden [Sch09]. Die Bekanntesten, beziehungsweise am weitesten Verbreiteten sind in der Tabelle 3.1 dargestellt.

Name	*Organisation*	*Ersthersteller*
EtherCAT	EtherCAT Technology Group (ETG)	Beckhoff
PROFINET	PROFIBUS Nutzerorganisation e.V. (PNO)	Siemens
Ethernet POWERLINK	Ethernet POWERLINK Standardization Group (EPSG)	B&R
Ethernet/IP mit CIP Sync	Open DeviceNet Vendor Association (ODVA)	Rockwell Automation
Modbus RTPS	Modbus-IDA	Schneider Electric
SERCOS-III	SERCOS International e.V. (SI)	-

Tabelle 3.1: Gängige Industrial Ethernet Varianten

3.1 Klassifizierung Industrial Ethernet

Um einen besseren Überblick zu erhalten, wird im Folgenden eine Kategorisierung durchgeführt. Diese besteht aus drei verschiedenen Klassen, deren Abstufung den Grad der Abweichung vom Ethernet Standard und der Netzwerkprotokolle, die zu erreichende Performance, sowie natürlich auch die Kosten wiederspiegelt. Diese Einteilung ist an jene von Martin Rostan [Ros08], beziehungsweise Prof. Dr.-Ing. Klaus Lebert [Leb08] angelehnt.

Zu bemerken ist noch, dass das in allen drei Klassen erwähnte Netzwerkprotokoll nicht zwangsläufig TCP/IP oder UDP/IP sein muss, jedoch wird bei allen in dieser Arbeit genauer betrachteten Industrial Ethernet Lösungen eines dieser Protokolle zur Verfügung gestellt.

3.1.1 Industrial Ethernet - Klasse 1

Zu dieser Kategorie zählen jene Lösungen, welche sowohl Standard Ethernet Hardware als auch den TCP(UDP)/IP-Software-Stack verwenden. Unter Standard Ethernet Hardware soll in dieser Arbeit zum herkömmlichen PC kompatible Hardware, bestehend aus MAC- und PHY-Schicht, verstanden werden. Methoden, die hier zur Optimierung eingesetzt werden, sind das Zeitscheibenverfahren und die Nutzung der maximalen Framelänge. Über der vierten Schicht wird ein für die Industrie-Automatisierung passender Applikationslayer aufgesetzt. Durch die Verwendung des gesamten Netzwerkprotokoll-Stacks wird die Performance hinsichtlich des Datentransfers geschmälert. Typische Werte für die Latenzzeit der Datenübertragung in dieser Kategorie liegen im Bereich von 10 ms bis 100 ms. Als typische Vertreter dieser Klasse gelten zum Beispiel Modbus-IDA und EtherNet/IP [J+07].

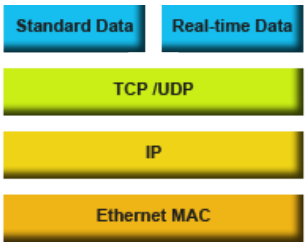


Abbildung 3.1: Kommunikations-Stack Industrial Ethernet - Klasse 1

3.1.2 Industrial Ethernet - Klasse 2

Bei den Ansätzen in dieser Kategorie sind die Layer 1 und 2 weitestgehend unmodifiziert nach dem Fast Ethernet Standard implementiert. Eventuell werden für eine Steigerung der Performance in Schicht 2 Anpassungen bei den Treiberbausteinen der eingesetzten Ethernet-Chips durchgeführt. Weiters werden zeit-kritische Daten mithilfe von VLAN Tags (IEEE 802.1Q) priorisiert und somit bevorzugt weitergeleitet. Kollisionen werden meist durch die Realisierung des Buses als Switched Ethernet und Betrieb im Voll-Duplex-Modus vermieden (siehe Abschnitt 2.3.1). Zur weiteren Optimierung der Gesamtlaufzeit wird in dieser Kategorie der TCP(UDP)/IP-Stack umgangen. Durch diese Maßnahmen können für Prozessdaten Latenzzeiten der Datenübertragung in der Größenordnung von 10 ms erreicht werden. Die beschriebenen Maßnahmen wurden zum Beispiel bei PROFINET RT umgesetzt.

3.1.3 Industrial Ethernet - Klasse 3

Hauptaugenmerk der Lösungen in dieser Kategorie liegt auf der bestmöglichen Performance. Die Ansätze dies zu erreichen sind durchaus unterschiedlich. Die am weitest verbreiteten

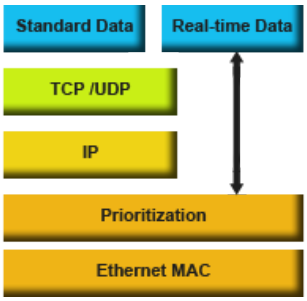


Abbildung 3.2: Kommunikations-Stack Industrial Ethernet - Klasse 2

Methoden sind das Zeitschlitzverfahren kombiniert mit dem Master/Slave-Prinzip, die Verwendung spezieller Switches, oder auch das Summenrahmenverfahren, auf welches später noch genauer eingegangen wird. Je nach Methode sind die vier unteren Schichten des OSI-Modells mehr oder weniger modifiziert. Möglich ist auch der Einsatz speziell angepasster Hardware in Form von FPGAs oder auch ASICs. All diese Modifikationen schließen jedoch den Einsatz von azyklischer Kommunikation, von zum Beispiel Parameterdaten, über TCP(UDP)/IP nicht aus. Die Latenzzeit der Datenübertragung kann in dieser Kategorie mit unter 1 ms angegeben werden. Ebenfalls in einem sehr kleinen Bereich, in etwa 1 µs, bewegt sich der Jitter dieser Lösungen. Eine genauere Betrachtung folgt in den Abschnitten 3.2 bis 3.4.

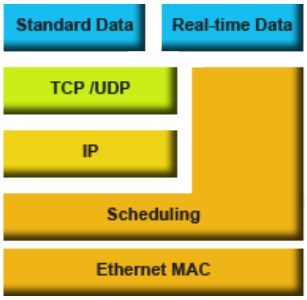


Abbildung 3.3: Kommunikations-Stack Industrial Ethernet - Klasse 3

Aktuell gibt es knapp 30 Lösungen in den Klassen 1 - 3 für Industrial Ethernet. Ein mit dieser Vielfalt entstehender Nachteil ist die Inkompatibilität zwischen den Implementierungen. Das bedeutet, eine Kommunikation zwischen zum Beispiel einem Ethernet/IP-Gerät und einem POWERLINK-Gerät ist ohne zusätzlichen Aufwand nicht möglich. Lösungen für derartige Anwendungen existieren zwar in Form von Proxys oder parallel betriebenen Kommunikations-Stacks, erhöhen aber den Aufwand sowie die Komplexität des Gesamtsystems und wirken sich keinesfalls positiv auf die Performance aus. Im weiteren Verlauf dieser Arbeit werden die Lösungen der Klassen 1 und 2 außer Acht gelassen, da für die in späteren Kapiteln betrachteten Anwendungsfälle eine Latenzzeit der Datenübertragung von über 1 ms nicht akzeptabel ist. Als Beispiel sei an dieser Stelle nur kurz die Kommunikation zwischen einem Regler und dem elektrischen Antrieb eines F1 Getriebeprüfstandes (PMM-Maschine mit einem Drehzahlgradient von circa 250 000 Umdrehungen pro Sekunde) erwähnt. Alleine eine Verzögerung der Steuerdaten von zum Beispiel 10 ms könnte eine

Drehzahländerung von rund 2500 Umdrehungen bedeuten, was durchaus einen Schaden an der Mechanik mit sich bringen kann. Zur Beschreibung der durchaus verschiedenen Lösungswege von Klasse 3 Industrial Ethernets, wird im Folgenden auf die Technologie von POWERLINK, PROFINET IRT und EtherCAT eingegangen.

3.2 POWERLINK

Ursprünglich entwickelt wurde dieses Klasse 3-Industrial Ethernet von der Firma Bernecker&Rainer und erstmals im November 2001 im Rahmen der SPS/IPC/DRIVES in Nürnberg vorgestellt. Bereits ein gutes Jahr später wurde der Standard offengelegt und das Standardisierungskonsortium Ethernet POWERLINK Standardization Group (EPSG) gegründet.

3.2.1 Aufbau und Funktionsweise

Ein POWERLINK-Netzwerk besteht aus einem Master - dem Managing Node (MN) - und mehreren Slaves - den Controlled Nodes (CN). Die Kommunikation läuft nach dem Schema des Zeitschlitzverfahrens gemischt mit Polling ab, in diesem Kontext auch "Slot Communication Network Management" (SCNM) genannt. Dabei wird ein Kommunikations-Zyklus in einen isochronen und einen asynchronen Übertragungszeitraum aufgeteilt (siehe Abbildung 3.4). Im isochronen Teil der Kommunikation werden die zeit-kritischen Daten in harter Echtzeit übermittelt. Jedem im Netzwerk befindlichen Controlled Node steht ein Zeitschlitz im gesamten Zyklus zur Datenübertragung zur Verfügung. Ausnahmen davon sind Controlled Nodes im Multiplex-Modus und solche die rein für eine azyklische Kommunikation konzipiert sind. Erstere können zum Beispiel I/Os darstellen, welche Signale von sich nur träge ändernden physikalischen Größen aufnehmen. In diesem Fall wäre es wenig sinnvoll den Bus in jedem Zyklus mit Daten zu "belasten", welche sich maximal alle - zum Beispiel - 100 Zyklen verändern. In solchen Fällen können sich mehrere Slaves mit ähnlichem Verhalten einen Zeitschlitz teilen, so dass jeder Controlled Node nur jeden n-ten Zyklus seine Daten aussendet beziehungsweise dazu aufgefordert wird. Die zweite Ausnahme stellen wie schon weiter oben erwähnt Slaves dar, welche gänzlich aus der zyklischen Kommunikation ausgeschlossen sind und lediglich nicht zeit-kritische Informationen bereit stellen. Da die azyklische Kommunikation standardmäßig nur dann stattfindet, wenn neben der zyklischen Übertragung noch eine Zeitreserve vorhanden ist, sollte bei vorhandenen, rein azyklischen Controlled Nodes darauf geachtet werden, dass die azyklische Kommunikation vom Managing Node regelmäßig durchgeführt wird [K+08].

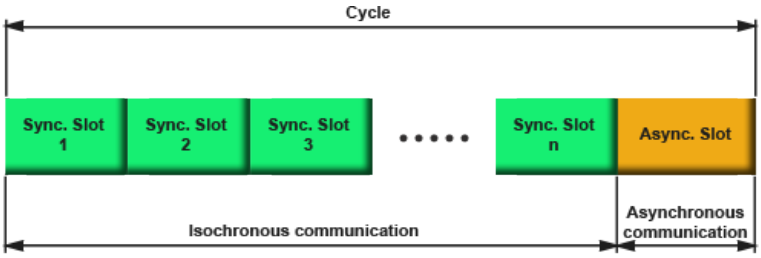


Abbildung 3.4: Ethernet POWERLINK Zeitschlitzverfahren

Die Synchronisation des Netzwerks, alle zeitlichen Abläufe und somit auch die Erteilung der Zugrifferlaubnis auf das Netzwerk an die verschiedenen Slaves wird vom Master gesteuert. In Abbildung 3.5 ist ein typischer Ablauf innerhalb eines Zyklus mit zugehörigen Anfragen vom Master und darauf folgenden Antworten der Slaves dargestellt. Jeder Controlled Node erhält sequentiell vom Managing Node das Recht auf den Buszugriff und darf nur dann Daten auf den Bus legen. Durch dieses Zugriffsverfahren werden Kollisionen gänzlich vermieden. Dadurch kann POWERLINK ohne weitere Modifikationen auf das CSMA/CD des 802.3u Fast Ethernet Standards aufbauen und in Folge dessen kann auch Standard-Hardware verwendet werden.

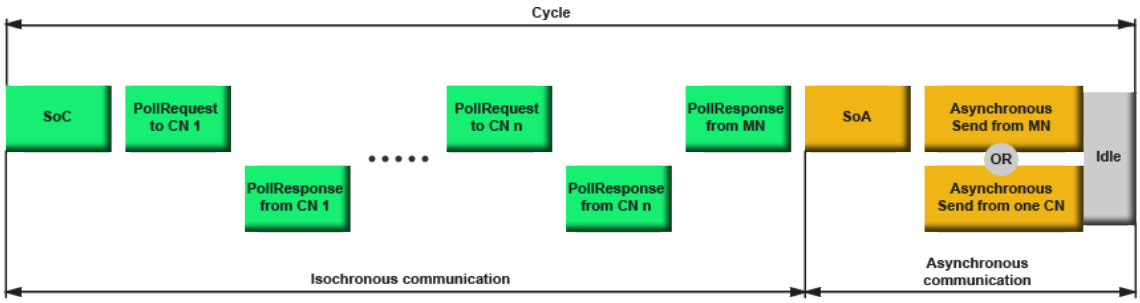


Abbildung 3.5: POWERLINK Zyklus

3.2.2 Adressierung der Slaves

Die Adressen der Controlled Nodes werden manuell über einen Knotenschalter an den jeweiligen Slaves eingestellt. Dabei steht eine Adresse im Umfang eines Oktetts zur Verfügung. Für Controlled Nodes ist der Bereich von 1 bis 239 reserviert (siehe Tabelle 3.2).

Knoten-ID	*Beschreibung*
0	Ungültige ID
1...239	Controlled Nodes
240	Managing Node
241...250	Reserviert
251	Pseudo-Node-ID für CNs, um sich selbst zu adressieren
252	Dummy Node
253	Diagnostic device
254	Ethernet Router
255	POWERLINK-Broadcast

Tabelle 3.2: POWERLINK Adressierung

Die IDs 251 bis 253 sind für Evaluierungsvorgänge gedacht. So kann ein Controlled Node mittels der Dummy-Knoten-ID als Zieladresse Nachrichten aussenden, ohne einen realen Knoten zu adressieren. Durch die Pseudo-Knoten-ID kann der jeweilige Slave zum Beispiel sein eigenes Poll-Response Frame kontrollieren.

3.2.3 POWERLINK im ISO/OSI-Referenzmodell

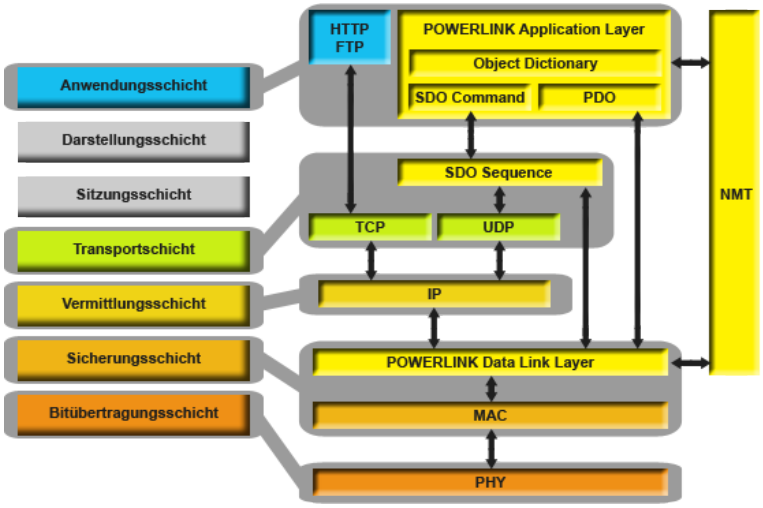


Abbildung 3.6: POWERLINK im ISO/OSI-Referenzmodell

In Abbildung 3.6 ist die Realisierung von POWERLINK in Bezug auf das ISO/OSI-Referenzmodell und dessen einzelne Schichten dargestellt. Man sieht, dass die physikalische sowie die MAC-Schicht dem Fast Ethernet Standard entspricht. In der physikalischen Ebene steht somit 100BASE-X (sowohl Kupfer als auch LWL) als Übertragungsmedium zur Verfügung. In der POWERLINK-Spezifikation wird die Verwendung von Halb-Duplex empfohlen.

In der Sicherungsschicht wurde zusätzlich ein POWERLINK Data Link Layer eingezogen. In dieser Schicht ist der Großteil der hier beschriebenen Funktionalität von POWERLINK spezifiziert. Dies reicht von der Definition der Funktionalität von Managing Node und Controlled Node, über die Abarbeitung der isochronen und asynchronen Kommunikation, bis hin zur Spezifikation eines Ethernet POWERLINK Frames.

Die über den 802.3u Ethernet Standard hinaus gehenden Spezifikationen - also jene ab der Vermittlungsschicht aufwärts - können in zwei Kategorien aufgeteilt werden:

- In den Schichten drei, vier und sieben sind die Protokolle IP (lt. RFC 791), TCP (lt. RFC 793), UDP (lt. RFC 768), HTTP (lt. RFC 2616) und FTP (lt. RFC 959) zur nicht zeit-kritischen Übertragung von Daten vorgesehen. In Kombination mit der POWERLINK Anwendungsschicht und dem Konzept der Service-Daten-Objekte (SDO) können im asynchronen Zeitfenster Service-Daten zwischen Managing Node und Controlled Node über zum Beispiel UDP/IP kommuniziert werden.

- Die Kommunikation in harter Echtzeit findet mittels dem Konzept der Prozess-Daten-Objekte (PDO) und des bereits zuvor beschriebenen Zeitschlitzverfahrens mit Polling statt. Wie man in Abbildung 3.6 erkennen kann, werden die isochron zu übermittelnden Prozessdaten direkt von der Anwendungsschicht an den POWERLINK Data Link Layer weitergereicht und dabei die Protokolle der Schichten drei und vier umgangen.

Hinter dem übergreifenden NMT-Block (Netzwerk-Management) verbirgt sich im Wesentlichen eine Zustands-Maschine welche zum Beispiel die Initialisierungsphase der Netzwerkteilnehmer steuert und somit das zyklische Verhalten der Kommunikationseinheiten bestimmt.

Generell sei noch angemerkt, dass das Konzept des "Object Dictionary" und der zugehörigen Prozess-Daten- und Service-Daten-Objekte nicht POWERLINK spezifisch ist. Diese Datenrepräsentation wurde aus der CANopen-Spezifikation EN 50325-4 entnommen und kann unter [CiA10] nachgelesen werden.

3.2.4 Telegramme und deren Verarbeitung

Bis auf eine Ausnahme werden alle POWERLINK-Nachrichten mittels Multicast-Adressierung ausgesendet. Die Ausnahme bezieht sich auf den Nachrichten-Typ Poll-Request - also die Erlaubnis für den Buszugriff - der per Unicast vom Managing Node an den jeweiligen Controlled Node gesendet wird. Alle anderen Nachrichten-Typen ("Start of Cycle", "Poll-Response", "Start of Asynchronous" und "Asynchronous Send") werden per Multicast-Adressierung ausgesendet. Somit können alle Teilnehmer am Bus alle Nachrichten mithören. Im Fall der SoC-Nachricht dient dies zur Synchronisation aller Teilnehmer. Die an alle ausgesendete Poll-Response-Nachricht eines Controlled Nodes kann somit auch, ohne den Master als Vermittler zu nutzen und somit mindestens einen Zyklus zu "verlieren", zur direkten Querkommunikation zwischen den Slaves verwendet werden.

In Abbildung 3.7 ist das Ethernet POWERLINK Basis-Frame dargestellt. Es zeigt die Integration des eigentlichen POWERLINK-Frames in den Ethernet II Datenbereich (siehe auch Abschnitt 2.3.2). Im sieben Bit langen Nachrichten-Typ-Feld wird eine der in Abbildung 3.5 dargestellten Nachrichten angekündigt. Die Ziel- und Quelladressen bestehen aus den in der Tabelle 3.2 aufgelisteten IDs. Das Datenfeld setzt sich je nach Nachrichtentyp aus unterschiedlichen Sub-Feldern zusammen. Diese sollen hier nicht weiter ausgeführt werden. Stattdessen wird auf die detaillierte Darstellung in [K+08], Seite 72 ff. verwiesen.

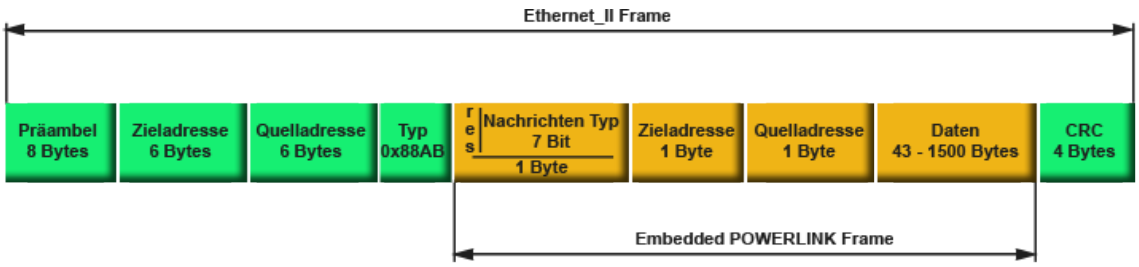


Abbildung 3.7: Ethernet POWERLINK Basis-Frame

3.2.5 Weitere Funktionalität des POWERLINK-Systems

Aus der Art der Adressierung der Controlled Nodes geht hervor, dass sich in einem POWERLINK Netzwerk-Segment nicht mehr als 239 Teilnehmer befinden, welche von einem Managing Node gesteuert werden. Der Zusammenschluss eines solchen Netzwerk-Segments findet üblicherweise mittels Hubs statt, welche bereits auf den POWERLINK Interface-Karten der einzelnen Slaves integriert sind. Somit kann ohne weiteren Hardware-Aufwand

eine logische Linien-Topologie erreicht werden. Um zum Beispiel eine Sterntopologie zu realisieren werden "externe" Hubs benötigt die, um die zusätzlich auftretenden Verzögerungszeiten und Jitter so gering als möglich zu halten, einem Klasse 2-Repeater entsprechen sollten. Eine Verzögerung von maximal 460 ns sowie zusätzlich möglicher Frame-Jitter von maximal 70 ns müssen bei der Auslegung eines solchen Netzwerkes berücksichtigt werden. Prinzipiell ist es auch möglich ein POWERLINK-Netzwerk mit Switches zu realisieren, jedoch entsprechen die zeitlichen Abläufe nicht mehr dem Standard.

3.3 PROFINET IRT

PROFINET (PROcess FIeld NET) stammt ursprünglich aus dem Hause Siemens und wird heute von der PROFIBUS Nutzerorganisation e.V. als offener Industrial Ethernet Standard betreut. Siemens selbst begann sehr früh sich mit dem Gedanken Ethernet im industriellen Umfeld anzufreunden. Bereits 1985 stellten sie "SINEC H1" als Ethernet für den Einsatz in industriellen Applikationen vor. Laut eigenen Angaben soll es sich, obwohl dieser Ansatz mit den heutigen Vorstellungen von Industrial Ethernet noch wenig gemeinsam hatte, um die Geburtsstunde von Industrial Ethernet handeln. Dabei ging es mehr um die Robustheit der Übertragung (niedrigere Rauschempfindlichkeit, verschraubbare Anschlüsse, anlagenweites Erdungskonzept), als um den Determinismus der Übertragung. Das erste Mal angekündigt wurde PROFINET von der PROFIBUS & PROFINET International bei einer Pressekonferenz im August 2000. Tatsächlich verfügbar und mit Echtzeit-Eigenschaften ausgestattet, war es dann im Jahr 2004.

Prinzipiell sind bei PROFINET, je nach Anforderung an das zeitliche Verhalten sowie natürlich auch dem Einsatzzweck, zwei unterschiedliche Realisierungen möglich. PROFINET CBA (Component Based Automation) wird in Anlagen eingesetzt, welche in kleinere, autonome Einheiten unterteilt werden können und zwischen denen eine Kommunikation - meist zwischen SPSen - mit einer Zykluszeit im Bereich von 100 ms (TCP/IP) bis 10 ms (Real-Time) gefordert wird. Die zweite PROFINET-Realisierung wird durch PROFINET I/O dargestellt. Wie der Name bereits andeutet geht es dabei um Kommunikation mit dezentralem I/O. Diese Lösung umfasst die Echtzeit-Kommunikation (RT, Real-Time) sowie die isochrone Echtzeit-Kommunikation (IRT, isochronous Real-Time) und erreicht Zykluszeiten von circa 1 ms. Zu erwähnen ist, dass nicht zwingend zwischen PROFINET CBA und PROFINET I/O entschieden werden muss, sondern auch ein paralleler Betrieb der beiden PROFINET-Arten möglich ist [PM08].

Da alle Betrachtungen in dieser Arbeit auf möglichst geringe Zykluszeiten ausgelegt sind, soll im Folgenden näher auf PROFINET I/O und dort im speziellen auf die IRT-Kommunikation eingegangen werden und die CBA-Variante weitestgehend außer Acht gelassen werden. PROFINET I/O ist in der IEC 61158 und IEC 61784 spezifiziert.

3.3.1 Aufbau und Funktionsweise

Vom Grundprinzip ist PROFINET dem zuvor vorgestellten POWERLINK ähnlich. Ein Übertragungszyklus wird in einen Abschnitt für Echtzeit-Kommunikation und einen für nicht-zeitkritische Datenübermittlung aufgeteilt (siehe Abbildung 3.8). Des Weiteren wird die Reihenfolge der Sendungen der teilnehmenden Geräte für die Echtzeit-Übertragung bereits bei der Netzwerkplanung festgelegt. Somit ist von Anfang an bekannt, wann welcher

Teilnehmer (im PROFINET-Kontext "I/O Device") welche Echtzeit-Daten sendet. Auch der Kommunikationspfad wird aus Optimierungsgründen in der Planung berücksichtigt. Im Gegensatz zu POWERLINK, wo die Kollisionen von Datenpaketen am Bus durch das Zeitschlitzverfahren mit Polling von vorn herein ausgeschlossen werden, werden bei PROFINET IRT die Kollisionen am Bus durch den Einsatz von Switches verhindert. Es handelt sich bei diesem Echtzeit-Ethernet um ein Switched Ethernet. Die Topologie kann entweder sternförmig über Multiport-Switches oder auch linienförmig über Zwei-Port-Switches, welche direkt im Ethernet-Controller am Gerät integriert sind, ausgeführt werden.

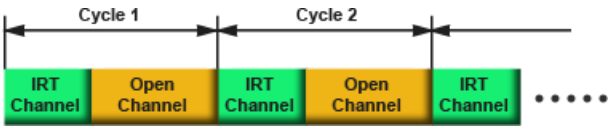


Abbildung 3.8: Aufteilung eines Buszyklus bei PROFINET IRT

Bei PROFINET gibt es die drei verschiedene Geräteklassen I/O-Controller, I/O-Supervisor sowie das I/O-Device. Der Controller wird meist durch eine SPS repräsentiert, in welcher die Applikation (z.B. Automatisierungsprogramm) hinterlegt ist. Bei isochroner Echtzeit-Kommunikation fungiert der I/O-Controller standardmäßig auch als Master-Clock. Ein PC oder auch ein HMI können als I/O-Supervisor verwendet werden. Zu dessen Aufgaben gehören unter anderem die Inbetriebsetzung der Anlage beziehungsweise der Kommunikation, sowie Diagnoseaufgaben. Das dezentrale I/O-Gerät wird bei PROFINET als I/O-Device bezeichnet und ist für das Lesen von den Eingängen beziehungsweise das Schreiben der Daten auf die Ausgänge zuständig.

Jedes PROFINET-Netzwerk umfasst mindestens einen I/O-Controller und ein oder mehrere I/O-Devices. Die Echtzeit-Kommunikation zwischen diesen Geräten kann bei PROFINET I/O in die RT-Klassen 1, 2 und 3 unterteilt werden. Da es in der vorliegenden Arbeit um kleinstmögliche Zykluszeiten mit geringem Jitter geht, sind die Klassen 1 und 2 für die weiteren Betrachtungen nicht relevant. Klasse 3 repräsentiert die Echtzeit-Variante PROFINET IRT und erreicht Zykluszeiten von 1 ms und knapp darunter mit einem maximalen Jitter von 1 µs. Die detaillierte Aufteilung eines PROFINET-Buszyklus ist in Abbildung 3.9 zu sehen. Der Großteil der im Folgenden erklärten Logik muss in den Switches implementiert sein. Diese sind für die korrekte Weiterleitung der jeweiligen Frames zum richtigen Zeitpunkt verantwortlich.

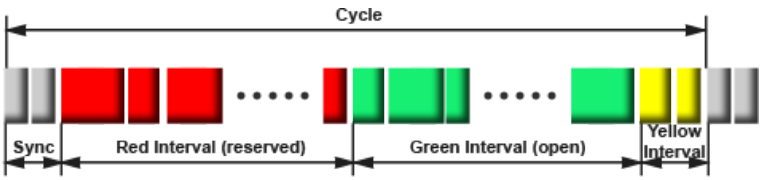


Abbildung 3.9: Die verschiedenen Intervalle innerhalb eines PROFINET IRT-Zyklus

Im ersten Teil des Zyklus erfolgt die Synchronisation aller Teilnehmer am Bus. Diese geht meist vom I/O-Controller aus, in welchem die Master-Clock integriert ist. Darauf folgt das sogenannte rote Intervall welches ausschließlich für die isochrone Echtzeit-Kommunikation (Klasse 3) vorgesehen ist. Die gesamte Kommunikation in diesem Ab-

schnitt muss bei der Planung des PROFINET-Busses festgelegt werden. Der zugehörige Planungsalgorithmus ist in der Norm IEC 61158 spezifiziert. Das bedeutet, dass nicht nur in den betreffenden Endteilnehmern der Sende- und Empfangszeitplan "ihrer" Frames hinterlegt sein muss, sondern dass auch in den Netzwerkkomponenten entlang des projektierten Kommunikationspfades das Weiterleiten der Frames definiert sein muss. Für jede Sendung ist hier auf Portebene definiert, welches Frame wann und mit welcher Länge gesendet werden muss. Einfluss auf das Timing nehmen hier die Netzwerktopologie, die Länge des jeweiligen Frames und auch die Länge des Verbindungsmediums zwischen den Kommunikationspartnern. Die Zuordnung der Frames zu ihrem Übertragungszeitraum erfolgt anhand ihrer Frame-ID. Sollten während des roten Intervalls UDP/IP-Daten, also nicht zeit-kritische Daten, von einem Switch empfangen werden, speichert dieser das Frame und sendet es im nächsten grünen Intervall aus. Optional kann nach dem Roten ein oranges Intervall für Klasse 2 Echtzeit-Kommunikation eingeplant werden. Das grüne Intervall steht für nicht zeit-kritische Kommunikation (zum Beispiel Diagnosedaten via UDP/IP) zur Verfügung. Dabei kann eine Priorisierung der verschiedenen Frames mittels der in IEEE 802.1Q spezifizierten VLAN-Tags erfolgen, wodurch die höher-prioren Daten von den Switches bevorzugt werden. Diese VLAN-Tags werden nur bei nicht zeit-kritischen Daten im grünen Intervall angewandt. IRT-Frames im roten Intervall werden ohne dieses Tag übermittelt. Als Übergang vom offenen zum reservierten Übertragungszeitraum dient das gelbe Intervall. In diesem werden von den Switches nur mehr Frames weitergeleitet, welche bis zum Ende des Intervalls vollständig transportiert werden können. Sollte dies zeitlich nicht möglich sein, werden die Daten zwischengespeichert und im nächsten grünen Intervall ausgesendet. Sollte während des offenen Intervalls ein IRT-Frame bei einem Switch eintreffen, wird dieses gelöscht und eine Alarmmeldung ausgegeben [Pro09].

3.3.2 Adressierung der Slaves

Die I/O-Devices werden im PROFINET IRT-Netzwerk über ihre MAC-Adresse, wie in der zweiten Schicht des ISO/OSI-Modells für Fast Ethernet spezifiziert, angesprochen. Dies hat zur Folge, dass sich die Ausdehnung des Netzwerkes auf ein Subnetz beschränkt, was aber für den Gewinn an Determinismus in Kauf genommen wird.

3.3.3 PROFINET IRT im ISO/OSI-Referenzmodell

In Abbildung 3.10 ist die PROFINET IRT-Spezifikation anhand des ISO/OSI-Referenzmodells dargestellt. Wie schon im Fall von POWERLINK ist zu sehen, dass für die Übertragung von Echtzeit-Daten der Kommunikations-Stack (TCP/IP, UDP/IP) umgangen wird. Dies führt zu einer erheblichen Performance-Steigerung zum Preis der eingeschränkten Netzwerkausdehnung. Die PHY- und MAC-Teilschichten bleiben unverändert zum 802.3u-Standard. Mit einer für PROFINET spezifizierten Bitrate von 100 MBit/s im Vollduplex-Modus und zusätzlicher Switched Ethernet-Technik mit zeitlich exakt projektiertem Datenverkehr, werden Kollisionen von vorn herein ausgeschlossen. Dadurch kann auch hier, wie schon zuvor bei POWERLINK, auf das CSMA/CD-Verfahren - und somit auf die unveränderte MAC-Schicht - aufgesetzt werden. Die nicht zeit-kritischen Daten, wie etwa Parameterdaten oder auch Statusinformationen, werden innerhalb der azyklischen Kommunikation (grüner Kommunikationsintervall, siehe Abbildung 3.9) über UDP/IP übertragen. Eben-

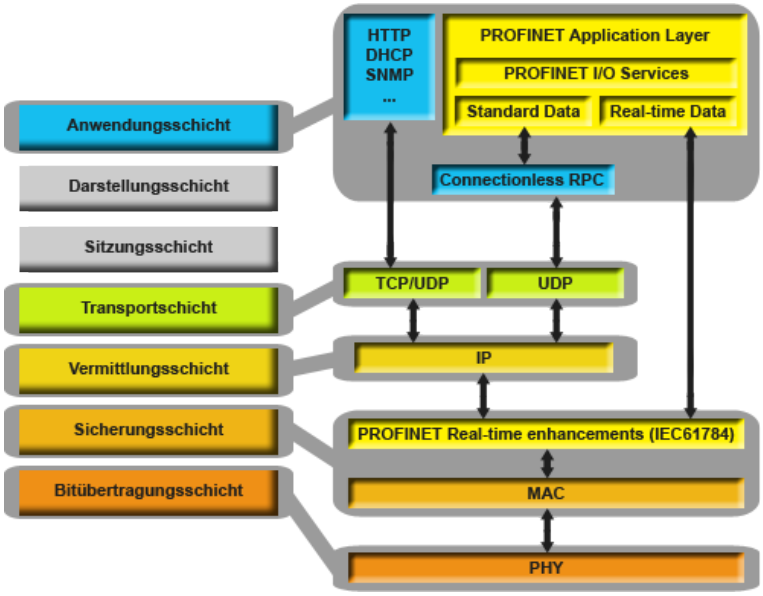


Abbildung 3.10: PROFINET IRT im ISO/OSI-Referenzmodell

falls ermöglicht wird die Kommunikation von herkömmlichen Ethernet-Daten über zum Beispiel HTTP und TCP/IP.

3.3.4 Telegramme und deren Verarbeitung

Die Frames bei PROFINET IRT werden je nach Art der zu übermittelnden Daten unterschiedlich aufgebaut. Hauptunterscheidungsmerkmal ist das zwei Byte lange Ethertype-Feld. Zum Beispiel wird für die Kommunikation von Echtzeit-Daten der Typ 0x8892, für Daten mit IP-Header 0x0800 und für Frames mit VLAN-TPID nach 802.1Q 0x8100 ins Typ-Feld eingetragen. In den Abbildungen 3.11 und 3.12 ist das Frame für die Übertragung von Standard-Daten (zum Beispiel Diagnosedaten) beziehungsweise jenes für die Übertragung von isochronen Echtzeit-Daten dargestellt.

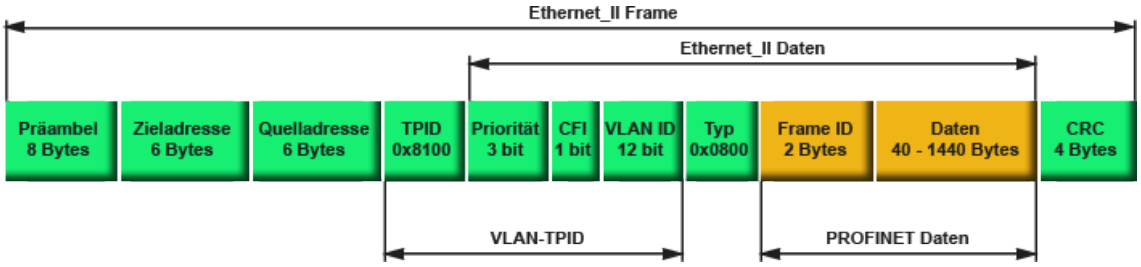


Abbildung 3.11: PROFINET IRT-Frame für die Kommunikation von nicht zeit-kritischen Daten (IP)

Beim Frame für die nicht zeit-kritische Datenübertragung kann man sehen, dass nach der MAC-Adresse des Absenders das TPID-Feld (Tag Protocol Identifier) den Typ des Ethernet II-Frames mit 0x8100 angibt und der betreffende Switch dadurch weiß, dass es sich

um keine Echtzeit-Daten handelt und die Weiterleitung in der Reihenfolge der Prioritäten zu erfolgen hat. Diese Priorität kann die Werte null bis sieben annehmen, wobei gilt, umso höher der Wert, desto höher die Priorität. CFI steht für "Canonical Format Indicator" und gibt das Format beziehungsweise die Bitreihenfolge der MAC-Adresse an. Für Ethernet gilt die Reihenfolge "LSB first" und wird in diesem Feld durch eine logische "0" repräsentiert. Darauf folgt das zwölf Bit lange VLAN ID-Feld welches das virtuelle Netzwerksegment identifiziert, zu dem das Frame gehört. In Abbildung 3.11 wird im zwei Byte langem Typ-Feld auf IP-Daten hingewiesen. Die tatsächlichen PROFINET-Daten werden von dem zwei Byte langem Frame-ID-Feld eingeleitet. Der Wert dieser ID kennzeichnet nicht nur das Frame selbst, sondern gibt durch den zugehörigen Wertebereich auch den Typ der Daten an. Zum Beispiel sind Werte zwischen 0x0100 und 0x7FFF für Klasse 3 Echtzeit-Daten reserviert und der Bereich von 0xC000 bis 0xFBFF für Daten via UDP/IP. Im Fall der Kommunikation von Nutzdaten werden diesen noch diverse Status-Informationen angehängt, auf deren Aufzählung an dieser Stelle verzichtet wird und stattdessen auf die Erklärungen unter [Fel09] verwiesen werden soll. Es sei nur soviel erwähnt, dass sich die minimale Anzahl an Datenbytes aufgrund der insgesamt vier Byte Status-Informationen auf 40 Byte verringert.

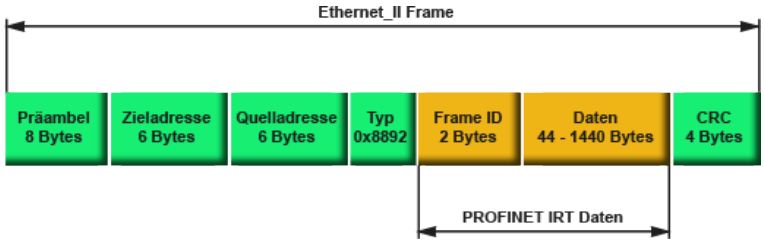


Abbildung 3.12: PROFINET IRT-Frame für die Kommunikation von isochronen Echtzeit-Daten

Im Fall von isochronen Echtzeit-Daten entfallen, wie in Abbildung 3.12 zu sehen ist, die VLAN TPID-Felder. Jeder Switch der ein solches Frame empfängt registriert den PROFINET IRT-Typ und leitet diesen anhand seiner Frame ID auf den in der Planungsphase festgelegten Kommunikationspfad weiter.

3.3.5 Weitere Funktionalität des PROFINET IRT-Systems

PROFINET-Geräte unterstützen standardmäßig Autonegotiation (siehe 2.3.1), das heißt die Teilnehmer am Bus stellen sich automatisch auf die bestmögliche Konfiguration bezüglich Bitrate und Duplex-Modus ein. Im Fall von PROFINET bedeutet das verpflichtend 100 MBit/s und Voll-Duplex. Weiters wurde aus Gründen der Vereinfachung betreffend der Verkabelung Auto-crossover für PROFINET spezifiziert. Dabei werden, falls nötig, die Sende- und Empfangsleitungen automatisch ausgekreuzt, was die Verwendung von unterschiedlichen Verbindungskabeln (gekreuzte beziehungsweise ungekreuzte Variante) überflüssig macht.

3.4 EtherCAT

Ursprünglich wurde dieses Echtzeit-Ethernet-Protokoll von der Firma Beckhoff entwickelt und erstmals bei der Hannover Messe im April 2003 vorgestellt. Es handelt sich, ebenso wie bei Powerlink, um einen offenen Standard, was bedeutet, dass diese Technologie kostenlos implementiert und verwendet werden darf. Des Weiteren stehen Guidelines zur Implementierung von Master und Slave, sowie Beispiel-Code von verschiedenen Seiten zur Verfügung (Auflistung auf der Homepage der EtherCAT Technology Group (ETG)). Ebenso befinden sich auch viele Master-Lösungen für die verschiedensten Betriebssysteme, sowie Slave-Lösungen im Bereich Aktuatorik & Sensorik, I/O-Systeme, hydraulische & pneumatische Systeme und viele andere am Markt. Mit Stand von April 2009 sind bereits 1000 Firmen Mitglieder der ETG, was ebenfalls auf die große Verbreitung dieses Standards hinweist.

3.4.1 Aufbau und Funktionsweise

Ein typischer Aufbau eines EtherCAT-Netzwerks beinhaltet einen Master und mehrere Slaves. Prinzipiell kann für den EtherCAT-Master, sowie für den EtherCAT-Slave Standard Ethernet Hardware verwendet werden. Um jedoch die Verzögerungszeit beim Weiterleiten der Daten im Slave gering zu halten und somit eine möglichst hohe Performance zu erzielen, wird die Slave-Seite mit spezieller Hardware (zum Beispiel ASICs oder FPGAs) implementiert. Als Übertragungsmedien kommen für EtherCAT alle im Abschnitt 2.3 dargestellten physikalischen Schichten in Frage. Durch die Kompatibilität zu Fast-Ethernet ist es somit möglich, zwischen den Teilnehmern die physikalische Schicht zu wechseln (zum Beispiel kann für weitere Strecken, beziehungsweise bei hoher EMV-Belastung von einer CAT5-Kupferlösung auf einen Lichtwellenleiter gewechselt werden). Neben 100BASE-TX beziehungsweise 100BASE-FX wurde für die Übertragung auf sehr kurzen Strecken (zum Beispiel innerhalb eines Klemmenblocks) der E-Bus spezifiziert. Dieser basiert auf LVDS-Übertragung nach IEEE P1596.3-1995.

EtherCAT unterstützt die verschiedensten Topologien, wobei die für Ethernet typische Stern-Topologie bei EtherCAT nur wenig praktische Bedeutung hat. Hauptgründe dafür sind der hohe Verkabelungsaufwand sowie die Verzögerungszeiten in den bei dieser Topologie benötigten Switches. Für hoch-performante Anwendungen wird - logisch gesehen - eine offene Ringstruktur verwendet, bei welcher der Master an einem offenen Ende das Telegramm an die Slaves sendet und am anderen das Antworttelegramm erhält. Aus physikalischer Sicht ergibt sich dadurch eine Strang-Struktur.

Aus Ethernet-Sicht stellt sich der EtherCAT-Bus als ein einzelner, großer Teilnehmer dar ("Summenrahmenverfahren[1]", siehe Abbildung 3.15). Hinter diesem verbergen sich eine Vielzahl von EtherCAT-Slaves, welche die einlaufenden Telegramme im Durchfluss verarbeiten, das heißt die für sie bestimmten Nutzdaten aus dem Telegramm entnehmen, beziehungsweise darin einblenden und dieses an den nächsten Teilnehmer weiterleiten. Der letzte EtherCAT-Slave leitet das von allen adressierten Teilnehmern bearbeitete Frame auf die Rückleitung um und sendet es somit wieder zurück zum ersten Slave, welcher das Frame als Antworttelegramm zurück zum Master sendet. Für diesen Signalfluss wird der

[1] Im Nutzdatenbereich eines herkömmlichen Ethernet-Frame nach IEEE 802.3 können mehrere EtherCAT-Telegramme platziert werden.

im Ethernet-Standard spezifizierte Vollduplex-Modus verwendet. Jeder Slave besitzt somit mindestens zwei Rx- und Tx-Schnittstellen, also ein Rx/Tx-Paar für die Hinleitung und Eines für die Rückleitung. Die Verarbeitung der Telegramme erfolgt immer in Hinrichtung. Die Rückleitung dient zum Lokalisieren und Schließen von Leitungsunterbrechungen, sowie zum Verstärken und Regenerieren des Signals.

Jeder EtherCAT-Slave erkennt automatisch, ob auf seiner Hin- beziehungsweise Rückleitung ein Trägersignal anliegt (siehe auch Carrier-Sense im Fast-Ethernet Standard unter 2.3). Aufgrund dieser Information werden die Signalpfade entsprechend geschaltet. Zum Beispiel erkennt der letzte Teilnehmer am EtherCAT-Bus, dass an seinem Tx-Anschluss der Hinleitung kein Trägersignal anliegt und schaltet diesen Anschluss auf den Rx-Anschluss der Rückleitung kurz, wodurch der Bus abgeschlossen ist. Um durch einen kurzzeitigen Ausfall eines nachfolgenden Teilnehmers diesen nicht dauerhaft auszuschließen, wird auch nach einem Kurzschluss weiter auf ein anliegendes Trägersignal geprüft und im positiven Fall der Abschluss wieder geöffnet. Durch diese automatische Erkennung ist eine exakte Lokalisierung eines eventuellen Leitungsbruchs möglich.

3.4.2 Adressierung der Slaves

Je nach Art der zu übertragenden Daten wird eine passende Adressierung der Teilnehmer am EtherCAT-Bus verwendet. Dabei kann grob in die Kommunikation von Parameterdaten, Diagnosedaten, sowie Prozessdaten unterteilt werden. In dieser Reihenfolge steigt auch die Anforderung an das zeitliche Verhalten. So werden Parameter azyklisch und relativ zeitunkritisch, Diagnosedaten ebenfalls azyklisch mit einer höheren zeitlichen Anforderung und Prozessdaten zyklisch mit einer vorgeschriebenen, maximalen Übertragungszeit kommuniziert.

Für die Übermittlung von Parameterdaten wird die physikalische Adressierung verwendet, wobei mit einem Telegramm ein Teil des bis zu 64 kByte großen Speichers eines Slaves angesprochen wird. Für diese Adressierung gibt es die zwei Varianten Auto-Increment- und Fixed-Adressierung. Die erst genannte wird vorwiegend in der Startup-Phase zur Verteilung von physikalischen Adressen ("Stationsadressen") verwendet. Der betreffende Teilnehmer wird anhand seiner Position im Kommunikationsring angesprochen. Falls der Master zum Beispiel den fünften Slave adressieren möchte, setzt er das 16 Bit Adressfeld (siehe auch 3.4.4) auf den Wert 0xFFFC ($\triangleq$-4_{dez}). Dieses Feld wird von jedem Teilnehmer - den das Telegramm durchläuft - um eins inkrementiert. Adressiert ist jener Slave, welcher ein Telegramm mit dem Wert Null im Adressfeld erhält. Bei der zweiten Variante, der Fixed-Address, wird der Teilnehmer über seine - in der wie oben beschriebenen Startup-Phase vergebenen - physikalischen Adresse angesprochen.

Für die Übertragung von Diagnosedaten werden Interrupts und Fixed-Adressierung, beziehungsweise Multiple-Adressierung verwendet. Jedes EtherCAT-Telegramm beinhaltet ein 16 Bit langes Interrupt-Feld mit dessen Hilfe von jedem Teilnehmer verschiedene Ereignisse an den Master gemeldet werden können. Tritt ein solches Ereignis auf, wird vom entsprechenden Teilnehmer im nächsten, empfangenen Telegramm eines oder mehrere der Interrupt-Bit gesetzt. Daraufhin wird der betreffende Slave über seine Stationsadresse angesprochen und so vom Master die Diagnosedaten abgeholt. Sollten innerhalb eines Zyklus von mehreren Teilnehmern Interrupt-Bit gesetzt werden, sendet der Master ein Telegramm mit Multiple-Adressierung aus um die gewünschten Informationen der Sla-

ves abzuholen. Dabei adressiert der Master den ersten Slave per Stationsadresse. Dieser setzt das Multiple-Read-Flag im Telegramm, wodurch die nachfolgenden Slaves erkennen können, dass sie ebenfalls adressiert sind. Jeder Teilnehmer der über den gewünschten physikalischen Adressbereich verfügt schreibt seine Stationsadresse und die angeforderten Informationen in das Datenfeld des Telegramms. Darüber hinaus stellen die ersten beiden Byte des Datenfeldes den Working-Pointer dar. Dieser zeigt auf das nächste freie Datum im Datenfeld und wird nachdem ein Teilnehmer seine Diagnose-Informationen im Datagramm eingetragen hat auch von diesem auf die nächste freie Stelle gesetzt.

Für die Kommunikation von zyklischen Prozessdaten wird die logische Adressierung verwendet. Dabei werden die Slaves nicht einzeln angesprochen, sondern über einen Teil eines bis zu 4 GByte großen, logischen Adressraums - das sogenannte Prozess-Image (PI) - welcher vom Master erzeugt und gehalten wird. Jeder Slave besitzt eine Fieldbus-Memory-Management-Unit (FMMU), deren Aufgabe es ist, während der Startup-Phase den vom Master ausgesendeten physikalischen Adressen eine logische Adresse zu zuordnen. Somit erhält jeder Slave während der Initialisierung (beziehungsweise dessen FMMU)

- eine logische, bitorientierte Startadresse
- die zugehörige Bitlänge
- eine physikalische Startadresse
- sowie den Typ (Input oder Output) des Speicherbereichs.

Wenn während des zyklischen Betriebes ein Teilnehmer von einem Telegramm (mit logischer Adressierung) durchlaufen wird, überprüft dessen FMMU dieses auf einen Adressmatch. Im positiven Fall werden die Daten - je nach Typ - aus dem entsprechenden Bereich des Datenfeldes entnommen, beziehungsweise darin eingeblendet. Somit stellt sich das über logische Adressierung kommunizierende EtherCAT-System beim Master als verteilter Speicher ("Distributed Memory") dar.

Als letzte Variante der Adressierung sei noch der Broadcast erwähnt. Dabei können physikalische Adressbereiche aller Slaves geschrieben, beziehungsweise verodert gelesen werden. Da sich ein EtherCAT-Slave immer in einem bestimmten Zustand befindet (Init / Safe-Operational / Pre-Operational / Operational) können zum Beispiel mit einem Broadcast-Read/-Write Zustandswechsel ausgelöst, beziehungsweise diese von allen Teilnehmern abgefragt werden.

3.4.3 EtherCAT im ISO/OSI-Referenzmodell

In Abbildung 3.13 ist die EtherCAT-Spezifikation anhand des ISO/OSI-Referenzmodells dargestellt. Wie schon bei den beiden zuvor aufgezeigten Echtzeit-Ethernets sieht man auch hier, dass der TCP/IP- beziehungsweise UDP/IP-Kommunikations-Stack für die zyklische Echtzeit-Kommunikation umgangen wird. Parameter- und Diagnosedaten können auch hier azyklisch ausgetauscht werden. Zu beachten ist, dass der dargestellte Zusammenhang zum Referenzmodell allgemein gültig und für eine EtherCAT-Master-Implementation zutreffend ist, jedoch sich bei Betrachtung eines Standard-Slaves die Sicherungsschicht leicht verändert darstellt. Wie schon erwähnt, werden als EtherCAT-Slaves eigene FPGAs beziehungsweise

ASICs verwendet. Bei der Implementation wird dort zum Beispiel auf das Buszugriffsverfahren CSMA/CD verzichtet, da es in einem reinen EtherCAT-Netzwerk nicht benötigt wird. Daher kann man im Fall der EtherCAT-Slaves die Sicherungsschicht als spezielle EtherCAT-MAC-Schicht sehen.

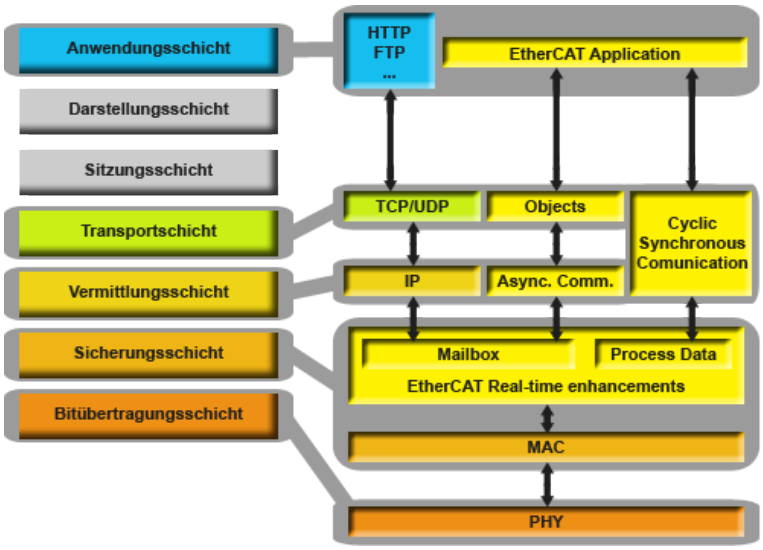


Abbildung 3.13: EtherCAT im ISO/OSI-Referenzmodell

3.4.4 Telegramme und deren Verarbeitung

In Abbildung 3.14 ist zu sehen, dass die Möglichkeit besteht einen IP- und UDP-Header ins Frame mit aufzunehmen. Dadurch entsteht der Vorteil, dass die im Ethernet-Frame verpackten EtherCAT-Kommandos auch über Subnetzgrenzen - also über Router - hinweg gesendet werden können. Diese Kodierung über UDP/IP ist vorwiegend für weniger zeitkritische Anwendungen gedacht, da die Verzögerungszeiten des Routers sowie jene der Software-Stacks zur Übertragungszeit hinzu kommen. Für alle in dieser Arbeit betrachteten Anwendungsfälle kommt ausschließlich das Ethernet-Frame mit speziellem Ether-Type zur Anwendung. Dies beschränkt zwar die Ausdehnung auf ein EtherCAT-Subnetz, jedoch mit dem klaren Vorteil einer höheren Performance. Im Ethernet-Typ-Feld wird für die UDP EtherCAT Kommunikation 0x8000 und für die Kommunikation in harter Echtzeit 0x88a4 eingetragen.

Wie zuvor erwähnt, verarbeiten die Teilnehmer die Telegramme im Durchlauf, wobei ein Slave für sich bestimmte Kommandos erkennt und diese ausführt. Die Verarbeitung selbst findet in Hardware statt und ist somit nicht von eventuellen Reaktionszeiten von nachgeschalteten Mikroprozessoren oder ähnlichem abhängig. Somit wird eine Verzögerung des Telegramms von wenigen Bitzeiten[2] pro Slave erreicht.

In Abbildung 3.15 ist ein Ethernet-Frame dargestellt, wie es auch in allen hier beschriebenen Anwendungen zum Einsatz kommt. Zu sehen ist, dass sich sämtliche EtherCAT-Informationen im Datenteil des Ethernet-Frames befinden. Im Gegensatz zu den voraus-

[2]Entsprechend des zugrunde liegenden 100 MBit/s Standards im Bereich von unter 500 ns.

gehenden Abbildungen wird bei den Darstellungen der EtherCAT-Frames auf die detaillierte Ausführung, zum Beispiel des IP-Headers, aus Platzgründen verzichtet. Details zum Ethernet-Header, sowie zum CRC sind im Abschnitt 2.3.2 zu finden. Ergänzend zu diesem Abschnitt sei noch angemerkt, dass ein jeder adressierter Teilnehmer das EtherCAT-Telegramm und somit auch das gesamte Ethernet-Frame verändert und somit die CRC-Summe von jedem Slave neu berechnet und ins CRC-Feld eingesetzt wird.

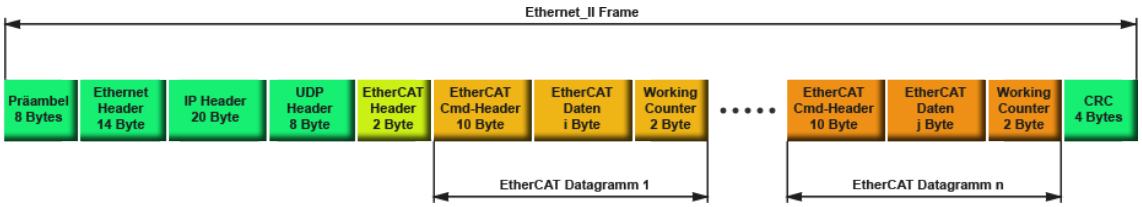


Abbildung 3.14: EtherCAT-Frame mit UDP/IP-Header

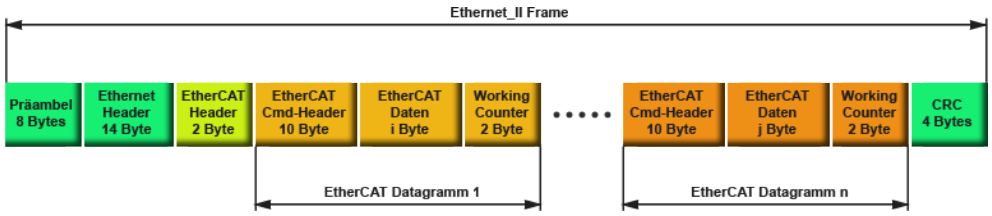


Abbildung 3.15: EtherCAT-Frame ohne UDP/IP-Header

Der zwei Byte lange EtherCAT-Header setzt sich zusammen aus (betrachtet von MSB nach LSB)

- einem vier Bit langem Typen-Feld (zum Beispiel weißt ”0001” auf ein folgendes EtherCAT-Kommando hin)
- einem reservierten Bit für spätere Implementationen
- einem elf Bit langem Längen-Feld (gibt die gesamte Länge der folgenden EtherCAT-Datagramme an)

Auf den EtherCAT-Header folgen ein oder mehrere Datagramme, wobei sich ein EtherCAT-Datagramm zusammensetzt aus (wiederum von MSB nach LSB)

- einem zehn Byte langem EtherCAT Command Header
  - Byte 1: Kommando (siehe Tabelle 3.3)
  - Byte 2: Index des Datagramms - wird von den Slaves unverändert weitergesendet und unterstützt den Master bei der Zuordnung der Antworttelegramms
  - Byte 3 & 4: Slave Adresse (ADP) - abhängig vom Kommando (Stationsadresse, logische Adresse, etc.).
  - Byte 5 & 6: Offset der Slave-Adresse (ADO) - wiederum abhängig vom verwendeten Kommando

  - Byte 7 & 8: Bit 0 bis 10 geben die Länge des folgenden Datenbereichs im Datagramm an. Bit 11 bis 15 sind Flag-Bit.
  - Byte 9 & 10: Interrupt-Feld (siehe Abschnitt 3.4.2)

- einem mindestens ein Byte langem Daten-Feld
- einem zwei Byte langem Working Counter-Feld, dessen Inhalt von jedem Teilnehmer, welcher adressiert war und das Datagramm erfolgreich bearbeitet hat, um eins inkrementiert wird. Dieses Feld bietet auch, neben der CRC-Summe, eine weitere Sicherheitsfunktionalität für den Master. Dieser kann anhand des enthaltenen Wertes überprüfen, ob das Telegramm von allen adressierten Teilnehmern erfolgreich bearbeitet wurde.

In Tabelle 3.3 sind die von den EtherCAT-Slaves unterstützten und in den ASICs/FPGAs implementierten Kommandos aufgeführt.

Kommando[3]	Slave Adresse (ADP)	Adressen-Offset (ADO)	Erklärung
NOP	-	-	Es wird keine Operation ausgeführt.
APRD, APWR, APRW	Enthält den Wert der inkrementellen Adressierung.	Enthält die physikalische Adresse der gewünschten Daten innerhalb des Slaves.	Lesen und/oder Schreiben eines physikalischen Bereichs mittels Auto-Increment-Adressierung.
FPRD, FPWR, FPRW	Enthält die Stationsadresse des gewünschten Slaves.	-	Lesen und/oder Schreiben eines physikalischen Bereichs mittels Fixed-Adressierung.
BRD, BWR, BRW	Alle Slaves sind adressiert. Dieses Feld wird von jedem durchlaufenen Teilnehmer inkrementiert.	-	Lesen (verodert) und/oder Schreiben eines physikalischen Bereichs aller Slaves.
LRD, LWR, LRW, LRO	Enthält das Hi-Word der logischen 32 Bit Adresse.	Enthält das Lo-Word der logischen 32 Bit Adresse.	Lesen und/oder Schreiben eines logischen Speicherbereichs (LRO ... verodertes Lesen).
MRD	Enthält die Stationsadresse des ersten adressierten Teilnehmers.	Enthält die physikalische Adresse der gewünschten Daten innerhalb des Slaves.	Lesen eines physikalischen Bereichs bei mehreren Slaves.

Tabelle 3.3: EtherCAT-Kommandos

3.4.5 Weitere Funktionalität des EtherCAT-Systems

Die Kommunikation zwischen zwei oder mehreren Slaves wird bei EtherCAT über den bis zu 4 GByte großen, logischen Adressraum realisiert. Dabei wird in diesem Adressraum Speicher für den Querverkehr reserviert und zyklisch vom Master ausgetauscht, das heißt im ersten Zyklus wird vom Slave, welcher den Input liefert, gelesen und dessen Daten in

den dafür vorgesehen Speicherbereich des Masters geschrieben. Im zweiten Zyklus sendet der Master das Datagramm mit den Informationen des ersten Slaves und einem LRD-Kommando an den zweiten Slave. Die zwischen den beiden Teilnehmern kommunizierten Daten sind für den Master transparent und werden von diesem lediglich zyklisch ausgetauscht.

Um einen synchronen Betrieb des Master und aller Teilnehmer zu garantieren, kann die Funktionalität der Distributed-Clocks verwendet werden. Es wird eine Clock der sich im EtherCAT-Netz befindenden Geräte zur Master-Clock erklärt (muss nicht zwangsläufig die des Masters sein) und die Clocks der restlichen Teilnehmer, sowie gegebenenfalls jene der Steuerung, darauf synchronisiert. Dazu wird von der Steuerung ein spezielles Telegramm ausgesendet, in welches der Teilnehmer mit der Master-Clock seine aktuelle Zeit einträgt und alle Anderen die Zeit aus diesem Telegramm entnehmen. Um die Laufzeit des Telegramms[4] zu berücksichtigen, wird vom Master ein Telegramm mit Broadcast-Read auf eine spezielle Adresse ausgesendet, worauf alle Slaves mit Distributed-Clock-Funktionalität ihre lokale Zeit (Hin- und Rückweg) in das Telegramm eintragen. Diese Informationen werden vom Master eingelesen und entsprechend ausgewertet [ETG08].

Weitere zu erwähnende Eigenschaften und Funktionalitäten wären zum Beispiel "Ethernet over EtherCAT", "CANopen over EtherCAT", die Möglichkeit der Anschaltung von herkömmlichen Feldbussen an das EtherCAT-System oder auch die verschiedenen Realisierungen von EtherCAT-Slave-Modulen. Da dies aber den Rahmen dieser Arbeit sprengen würde, wird an dieser Stelle auf weitere Ausführungen verzichtet.

3.5 Vergleich der Echtzeit-Ethernet Varianten

In vielen Bereichen sind die erzielten Leistungen der vorgestellten Industrial Ethernet Lösungen ähnlich und auch große Teile der jeweiligen Spezifikationen decken sich. So wird im Bereich der physikalischen Ebene bei allen Lösungen 100BASE-TX unterstützt und auch Großteils in der Praxis verwendet. Die Anschlüsse sind für diese Ethernet-Ausführung bei allen Lösungen gleich. Für den herkömmlichen Einsatz werden RJ-45-Stecker und für robustere Anforderungen (IP67-Konform) M12-Anschlüsse verwendet.

Für die Geräte-Beschreibungen, egal ob diese als Slave, I/O-Device oder auch als Controlled Node bezeichnet werden, setzen alle Lösungen auf XML-basierte Beschreibungsdateien. Im Fall von POWERLINK und EtherCAT werden direkt XML-Dateien eingesetzt. Bei PROFINET werden diese als GSD-Dateien (General Station Description) bezeichnet, welche jedoch ebenfalls auf dem XML-Standard basieren. Der große Vorteil dieser Beschreibungsdateien ist der minimierte Konfigurationsaufwand eines Industrial Ethernets, auch wenn Geräte von verschiedenen Herstellern im Netzwerk integriert sind. Diese Dateien geben dem Master Auskunft darüber, wie der Zugriff auf relevante Geräteinformationen (zum Beispiel Status, Eigenschaften oder auch Parametrierung des Teilnehmers) zu erfolgen hat.

Im Bereich der Synchronisation sind qualitativ ebenfalls keine großen Unterschiede zu sehen. Alle Lösungen schaffen eine Synchronisation der Kommunikation im Bereich von 1 µs und darunter. Ob diese Synchronisierung mittels "Distributed Clocks" (EtherCAT), nach IEEE1588 (PROFINET IRT) oder auch Frame-basiert (POWERLINK) stattfindet, ist an

[4] Verzögerung innerhalb der einzelnen Slave auf dem Hin- und Rückweg, sowie der Übertragungsstrecke selbst.

dieser Stelle nicht relevant. Wichtig ist lediglich, dass die technologischen Voraussetzungen bestehen und in dem genannten Bereich liegen.

Die tatsächliche Verwendung von Standard-Ethernet-Hardware ist nur bedingt möglich. Hier kommt es, zumindest bei den Lösungen POWERLINK und EtherCAT, sehr stark auf das geplante Einsatzszenario und die dadurch geforderte Performance an. So können bei POWERLINK unter Verwendung eines Standard Onboard Ethernet Controllers Zykluszeiten im Bereich von 500 µs und ein Jitter von circa 30 µs erreicht werden. Bessere Performance kann mit dem Einsatz einer speziellen Master-PCI-Karte erreicht werden. Der größte Vorteil dort ist ein Co-Prozessor, welcher die Verarbeitung des Protokoll-Stacks übernimmt und so die CPU entlastet. Erzielbare Zykluszeiten liegen im Bereich von 100 µs mit einem Jitter von 0.1 µs [EPS08]. Bei EtherCAT verhält es sich, obwohl im Vergleich zu POWERLINK leicht bessere Ergebnisse mit einem Standard-Ethernet-Controller erzielt werden können, ungefähr gleich. Auf der Seite der Slaves wird bei hohen Anforderungen an die Performance bei allen Lösungen auf Spezial-Hardware in Form von FPGAs und ASICs gesetzt. POWERLINK integriert dort die nötigen Hubs und PROFINET die benötigten Cut-Through Switches. EtherCAT benötigt diese Spezial-Hardware um die EtherCAT-MAC-Schicht zu integrieren, welche das unter Abschnitt 3.4 erwähnte Ein- und Ausblenden der Daten im "durchlaufenden" Frame möglich macht. Das bedeutet, bei POWERLINK und PROFINET sind Slaves nach dem Fast Ethernet Standard prinzipiell möglich, wohin gegen ein EtherCAT-Slave ohne die erwähnten Maßnahmen nicht funktioniert.

Ein eventueller Nachteil bei PROFINET, gegenüber den beiden anderen Lösungen, ist der Mehraufwand in der Planungsphase, da alle zeitlichen Abläufe und Kommunikationspfade der isochronen Echtzeit-Kommunikation bereits bei der Projektierung des Netzwerkes geplant werden müssen. Weiters muss der Planungsalgorithmus auch bei einer Änderung der Netzwerkkonfiguration neu durchlaufen werden. Dieser "Nachteil" ist aber eher Geschmackssache und somit soll im Folgenden auf den vermutlich größten Unterschied zwischen den vorgestellten Echtzeit-Ethernets näher eingegangen werden.

3.5.1 Vergleich der Zykluszeit - Methodik

Eine der wichtigsten Kenngrößen, wenn nicht die Wichtigste, stellt die minimal erreichbare Zykluszeit eines Industrial Ethernets dar. Aus diesem Grund sollen in diesem Abschnitt die minimalen Zykluszeiten der aufgezeigten Lösungen dargestellt werden. Als Zykluszeit soll jene Zeit verstanden werden, die der Controller für einen vollständigen Austausch von Eingangs- beziehungsweise Ausgangsdaten mit allen Teilnehmern am Bus benötigt. Es wird in diesem Abschnitt nur die minimal erreichbare Zykluszeit bei der Kommunikation in harter Echtzeit betrachtet. Eventuelle azyklische Parameter- beziehungsweise Diagnosedatenübertragungen werden nicht berücksichtigt, da der Fokus auf der schnellst möglichen Kommunikation liegt. Die Vorgehensweise ist an jene von [Pry08] und [J+07] angelehnt. Aufgrund seiner Bedeutung, sowohl in der Automatisierungsindustrie generell als auch in den praktischen Ausführungen dieser Arbeit im speziellen, sollen die folgenden Berechnungen auf Basis der Linien-Topologie erfolgen. Es muss in den Abschätzungen darauf geachtet werden, dass die jeweils angegebene minimale Menge an Nutzdaten nicht unterschritten wird. Sollte dies der Fall sein, muss mit der Mindestlänge des Ethernet II-Frames gerechnet werden. Die in 3.5.2 - 3.5.4 aufgestellten Zusammenhänge sollen als Abschätzung verstanden werden um qualitative Aussagen bezüglich des zeitlichen Verhaltens treffen zu

können.

Im Folgenden sollen die Bezeichnungen Master und Slave stellvertretend für die tatsächlich gebräuchlichen, lösungsspezifischen Bezeichnungen gültig sein. Des Weiteren werden folgende Abkürzungen verwendet:

- T_{M_fwd} ... Zeit, welche der Master zum Weiterleiten des Pakets benötigt
- T_{S_fwd} ... Zeit, welche ein Slave zum Weiterleiten des Pakets benötigt
- T_{PHY} ... Durch die physikalischen Schichten hervorgerufene Verzögerung (Senden und Empfangen)
- T_C ... Durch das Übertragungsmedium bedingte Verzögerungszeit
- n_{Slaves} ... Anzahl der Slaves
- $d_{Payload}$ ... Nutzdaten pro Slave in Byte
- B ... Verwendete Bandbreite in $\frac{Bit}{s}$
- $d_{Ethernet\_Max\_Length} = 1538\,Byte$ ... Maximale Anzahl der Byte in einem Ethernet II-Frame inklusive Header und Interframe-Gap
- $d_{Ethernet\_Header} = 38\,Byte$ ... Header-Daten pro Ethernet-Frame inklusive Interframe-Gap

3.5.2 Zykluszeit - POWERLINK

Ethernet POWERLINK arbeitet auf Grund der verwendeten Hubs im Halb-Duplex-Modus. Für die Abschätzung der minimalen Zykluszeit bedeutet das, dass sowohl der Kommunikationspfad vom Master zu den Slaves als auch der umgekehrte Weg eingerechnet werden muss. Für die Betrachtung der Zykluszeit von POWERLINK sind die folgenden, erweiternden Definitionen notwendig:

- $d_{POWERLINK\_Min\_Payload} = 43\,Byte$ ... Minimal zu übertragende Anzahl an Nutzdaten
- $d_{POWERLINK\_Header} = 3\,Byte$ ... Header-Daten je POWERLINK-Frame

Die Angaben über das zeitliche Verhalten der einzelnen Netzwerkelemente wurden vorwiegend aus [Kra08] übernommen:

- $T_{Hub\_fwd} = 500ns$ ... Die Verzögerungszeit für die Weiterleitung in einem Hub bei Linientopologie
- $T_{S\_fwd} \approx 1\mu s$ ... Zeit, welche ein Slave zum Weiterleiten des Pakets benötigt (FPGA-Implementierung)
- $T_{PHY} \approx 500ns$ ... Summe der Verzögerung der physikalischen Schichten (Senden und Empfangen)
- $T_C \approx 5\frac{ns}{m}$

Die Verzögerung der Weiterleitung im Master ergibt sich mit den obigen Definitionen folgendermaßen:

$$T_{M\_fwd} = \frac{n_{Slaves} \cdot \left(d_{Payload} + d_{Ethernet\_Header} + d_{POWERLINK\_Header}\right) \cdot 8}{B} \tag{3.1}$$

Der Halb-Duplex-Modus von POWERLINK wird mittels des Faktors "2" bei der Weiterleitungs-Verzögerung des Masters berücksichtigt. Dadurch ergibt sich für ein POWERLINK-Netzwerk in Linientopologie

$$T_{POWERLINK\_Min\_Cycle} = 2 \cdot T_{M\_fwd} + T_{PHY} + n_{Slaves} \cdot \left(T_{Hub\_fwd} + T_C + T_{S\_fwd}\right) \tag{3.2}$$

3.5.3 Zykluszeit - PROFINET IRT

Um den realen Implementierungen gerecht zu werden, wurde in den Abschätzungen der Zykluszeit für PROFINET IRT der sogenannte "Slipstreaming effect" berücksichtigt. Dabei wird die gesamte Zykluszeit verringert, indem der Master das Frame an den am weitest entfernten Slave als erstes sendet. Weiters wird der Voll-Duplex-Modus insofern berücksichtigt, dass nur die Zeit für die Kommunikation vom Master zum Slave gezählt wird. Für die Betrachtung der Zykluszeit von PROFINET IRT sind die folgenden, erweiternden Definitionen notwendig:

- $T_{Switch\_fwd} = T_{S\_fwd}$ ... Die Verzögerungszeit für die Weiterleitung in einem (integrierten) Switch wird mit jener eines Endknoten gleichgesetzt
- $d_{PROFINET\_Min\_Payload} = 40\,Byte$ ... Minimal zu übertragende Anzahl an Nutzdaten
- $d_{PROFINET\_Header} = 6\,Byte$ ... Header-Daten je PROFINET-Frame

Die Angaben über das zeitliche Verhalten der einzelnen Netzwerkelemente wurden vorwiegend aus [Pry08] übernommen:

- $T_{S\_fwd} \approx 3\mu s$ ... Summe der Verzögerung für die Weiterleitung eines Frames in einem (integrierten) Cut-Through Switch[5]
- $T_{PHY} \approx 500ns$ ... Summe der Verzögerung der physikalischen Schichten (Senden und Empfangen)
- $T_C \approx 5\frac{ns}{m}$

Die Verzögerung der Weiterleitung im Master ergibt sich mit den obigen Definitionen zu

$$T_{M\_fwd} = \frac{n_{Slaves} \cdot \left(d_{Payload} + d_{Ethernet\_Header} + d_{PROFINET\_Header}\right) \cdot 8}{B} \tag{3.3}$$

[5]Der Cut-Through Switch achtet bei einem eintreffenden Frame lediglich auf die MAC-Adresse und leitet das Paket entsprechend weiter. Aus Performance-Gründen findet somit auch keinerlei Fehlerüberprüfung statt.

Durch Berücksichtigung des "Slipstreaming effects" muss in der Abschätzung der gesamten, minimalen Zykluszeit nur die Verzögerungszeit der Paket-Weiterleitung des Masters mit der Anzahl der Pakete multipliziert werden. Dadurch ergibt sich für ein PROFINET-Netzwerk in Linientopologie

$$T_{PROFINET\_Min\_Cycle} = T_{M\_fwd} + T_{PHY} + T_C + T_{S\_fwd} \tag{3.4}$$

3.5.4 Zykluszeit - EtherCAT

Für die Betrachtung der Zykluszeit von EtherCAT sind die folgenden, erweiternden Definitionen notwendig:

- $d_{EtherCAT\_Max\_Payload} = 1498\,Byte$ ... Maximale Anzahl der Byte im Nutzdatenbereich inklusive EtherCAT-Header (siehe Abbildung 3.15)
- $d_{EtherCAT\_Min\_Payload} = 32\,Byte$ ... Minimal zu übertragende Menge an Nutzdaten
- $d_{EtherCAT\_Telegram\_Header} = 12\,Byte$ ... Header-Daten pro EtherCAT-Telegramm
- $d_{EtherCAT\_Header} = 2\,Byte$ ... Header-Daten pro EtherCAT-Frame
- $d_{EtherCAT\_Payload\_Part\_Frame} = (n_{Slaves} \cdot d_{Payload})\, mod\, (d_{EtherCAT\_Max\_Payload} - n_{Slaves} \cdot d_{EtherCAT\_Telegram\_Header})$ ... Nutzdaten im Frame mit nicht vollständig ausgefülltem Nutzdatenbereich
- $n_{EtherCAT\_Telegrams\_Part\_Frame}$ ... Anzahl der EtherCAT-Telegramme im Frame mit nicht vollständig ausgefülltem Nutzdatenbereich
- $d_{Header\_Data\_Part\_Frame} = d_{Ethernet\_Header} + d_{EtherCAT\_Header} + n_{EtherCAT\_Telegrams\_Part\_Frame} \cdot d_{EtherCAT\_Telegram\_Header}$ ... Byte an Header-Daten im Frame mit nicht vollständig ausgefülltem Nutzdatenbereich
- k_{Frames} ... Anzahl der Frames pro Zyklus
- $k_{Frames} - 1 = \left\lfloor \frac{(n_{Slaves} \cdot d_{Payload})}{(d_{EtherCAT\_Max\_Payload} - n_{Slaves} \cdot d_{EtherCAT\_Telegram\_Header})} \right\rfloor$ ... Anzahl der Frames mit ausgefülltem Nutzdatenbereich
- $T_{full\_Frame} = \frac{d_{Ethernet\_Max\_Length} \cdot 8}{B}$ ... Zeit für die Weiterleitung eines Frames mit ausgefülltem Nutzdatenbereich
- $T_{part\_Frame} = \frac{(d_{EtherCAT\_Payload\_Part\_Frame} + d_{Header\_Data\_Part\_Frame}) \cdot 8}{B}$ ... Zeit für die Weiterleitung des Frames mit nicht vollständig ausgefülltem Nutzdatenbereich

Die Angaben über das zeitliche Verhalten der einzelnen Netzwerkelemente wurden vorwiegend aus [Pry08] übernommen:

- $T_{S\_fwd} \approx 450ns$ ... Summe der Verzögerung für die Weiterleitung (Hin- und Rückleitung)
- $T_{S\_PHY} \approx 500ns$ ... Summe der Verzögerung der physikalischen Schichten (Senden und Empfangen)

- $T_C \approx 5\frac{ns}{m}$

Die Verzögerung der Weiterleitung im Master ergibt sich mit den obigen Definitionen zu

$$T_{M\_fwd} = (k_{Frames} - 1) \cdot T_{full\_Frame} + T_{part\_Frame} \tag{3.5}$$

Und die minimale Zykluszeit für ein EtherCAT-Netzwerk in Linientopologie ergibt sich zu

$$T_{EtherCAT\_Min\_Cycle} = T_{M\_fwd} + T_{PHY} + n_{Slaves} \cdot (T_{S\_fwd} + T_C) \tag{3.6}$$

3.5.5 Anforderungen am Prüfstand

Klares Ziel ist es, bis zu fünf E-Maschinen (aktive Dynamometer[6] - kurz "Dynos") über einen schnellen und effizienten Bus mit einer Steuereinheit beziehungsweise dem Automatisierungssystem zu verbinden. Zudem muss es möglich sein, weitere Knoten wie etwa Echtzeit-Simulatoren, in dieses Netzwerk zu integrieren. Die minimale Frequenz der zyklischen Echtzeit-Daten beträgt definitionsgemäß 10 kHz und soll später auf 20 kHz gesteigert werden. Die Verzögerungszeiten am Bus selbst, sind daher so gering wie möglich zu halten und sollte den in Abbildung 3.16 dargestellten Kommunikationspfad in maximal zwei Zyklen ermöglichen. Weiters sind in Tabelle 3.4 die pro Dyno minimal zu übertragenden Daten dargestellt. Daraus ergibt sich als Summe an Sendedaten pro Dyno 14 Byte sowie 22 Byte an Empfangsdaten. Aufgerechnet auf eine maximale Konfiguration mit fünf Dynos würde dies 70 Byte Sendedaten und 110 Byte Empfangsdaten ergeben. Diese beiden Werte sollen als Grundlage für die unter 3.5.6 durchgeführte Analyse dienen.

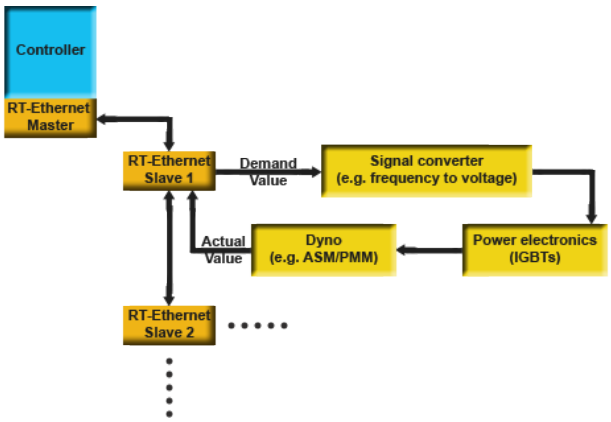


Abbildung 3.16: Kommunikationspfad am Prüfstand

3.5.6 Analyse der Prüfstandsanforderung

Mit dem im vorausgegangenen Abschnitt dargestellten Umfang an Sende- und Empfangsdaten, soll mit Hilfe der in den Abschnitten 3.5.2 bis 3.5.4 aufgestellten Zusammenhän-

[6] Als aktiver Dyno wird eine E-Maschine bezeichnet, welche nicht nur als Last fungiert und somit zum Beispiel die Leistung einer VKM aus Messung des Drehmoments und der Drehzahl bestimmen kann, sondern auch als Antrieb verwendet werden kann.

Beschreibung	*Byte*	*Master ist*
Elektrisches Stellmoment	4	Sender
Kontroll-Wort zum Umrichter	2	Sender
Reserviert für spätere Implementationen	4	Sender
Reserviert für spätere Implementationen	4	Sender
Aktuelles Drehmoment	4	Empfänger
Aktuelle Drehzahl	4	Empfänger
Aktueller Rotorwinkel	4	Empfänger
Status-Wort vom Umrichter	2	Empfänger
Reserviert für spätere Implementationen	4	Empfänger
Reserviert für spätere Implementationen	4	Empfänger

Tabelle 3.4: Nutzdaten pro E-Maschine

ge, eine Abschätzung der Zykluszeiten der drei aufgezeigten Lösungen durchgeführt werden. Vereinfachend wird angenommen, dass in beide Richtungen gleich große Pakete mit 110 Byte an Nutzdaten kommuniziert werden und diese gleichmäßig auf fünf Teilnehmer zu je 22 Byte Sende- beziehungsweise Empfangsdaten aufgeteilt werden. Weiters wird eine Übertragungsstrecke von zehn Meter zwischen Master und erstem Slave, sowie ebenfalls zehn Meter zwischen den einzelnen Slaves angenommen. Daraus ergibt sich eine Gesamtdistanz von 50 Meter zwischen dem Master und dem am weitesten entfernten Slave. Beispielhaft soll die Berechnung der beschriebenen Konfiguration für die Variante EtherCAT durchgeführt werden. Alle anderen Berechnungen erfolgen analog dazu mit Hilfe von Matlab. Die bekannten Werte werden somit für EtherCAT in die Gleichungen 3.5 und 3.6 eingesetzt. Daraus ergibt sich

$$T_{M\_fwd} = (k_{Frames} - 1) \cdot T_{full\_Frame} + T_{part\_Frame}$$

$$= 0 \cdot T_{full\_Frame} + \frac{(110 + 100) \cdot 8}{100 \cdot 10^6} = 16.8\,\mu s$$

und die zu erreichende minimale Zykluszeit in dieser Konfiguration zu

$$T_{EtherCAT\_Min\_Cycle} = T_{M\_fwd} + T_{PHY} + n_{Slaves} \cdot (T_{S\_fwd} + T_C)$$

$$= 16.8 \cdot 10^{-6} + 0.5 \cdot 10^{-6} + 5 \cdot (0.45 \cdot 10^{-6} + 5 \cdot 10^{-9} \cdot 10) = 19.8\,\mu s$$

Ein Vergleich der minimalen Zykluszeiten laut einer Konfiguration nach Abschnitt 3.5.5 ist in Abbildung 3.17 dargestellt. Es ist zu sehen, dass bei einer niedrigen Nutzdatenlast pro Slave, die Technik des Summenrahmenverfahren und somit EtherCAT klar im Vorteil ist. Zu dieser Berechnung muss noch angemerkt werden, dass auf Grund der Mindestlänge des Ethernet II-Frames im Fall von POWERLINK, anstatt der 22 Byte, 43 Byte und bei PROFINET 40 Byte im Bereich der Nutzdaten übermittelt werden.

Die aufgestellten Zusammenhänge in den Gleichungen 3.1 bis 3.6 sind für Anlagen mit Linientopologie allgemein gültig und somit kann mit deren Hilfe die minimal erreichbare Zykluszeit auch für viele andere Szenarien abgeschätzt werden. Zum Abschluss dieses Kapitels

zeigt die Abbildung 3.18 den Vergleich zwischen den drei vorgestellten Echtzeit-Ethernet-Varianten in Bezug auf eine variable Anzahl von Slaves. Es werden 50 Byte Nutzdaten je Kommunikationsrichtung und eine Distanz von zehn Metern zwischen den Teilnehmern verwendet. Die Anzahl der Slaves variiert von eins bis 20. Man sieht, dass sich der Vorteil des Summenrahmen-Frames hält und die Zykluszeiten auseinander laufen. Bei 20 Teilnehmern sendet der EtherCAT-Master immer noch ein einziges Frame (mit 1000 Byte an Nutzdaten), während die anderen Lösungen einzelne Frames an jeden Slave senden.

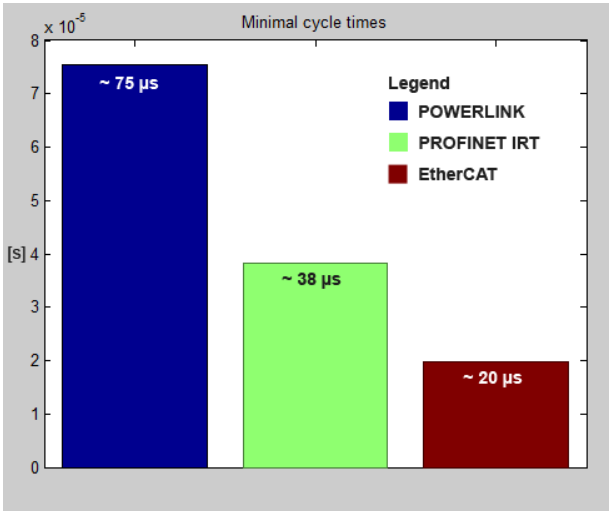


Abbildung 3.17: Vergleich der minimalen Zykluszeiten anhand einer realen Prüfstandskonfiguration

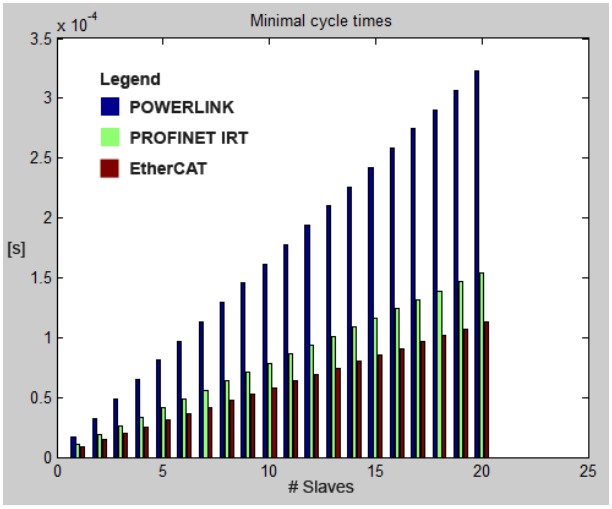


Abbildung 3.18: Vergleich der minimalen Zykluszeiten mit variabler Anzahl an Slaves

Kapitel 4

Einsatz von Echtzeitkommunikation

In diesem Kapitel soll der Aufbau des Testsystems, sowohl aus Hard- als auch aus Softwaresicht, dargestellt werden. Die Beschreibung der Hardware soll einen Einblick in die Planungsaktivitäten des Systems geben. Die Erklärungen bezüglich der Software sollen eine Übersicht, angefangen mit den verwendeten Betriebssystemen, bis hin zu den adaptierten beziehungsweise neu erstellten Simulationsmodellen geben. Des Weiteren werden die verschiedenen Testszenarien und deren Resultate beschrieben.

4.1 Testsystem - Hardware

Die Anforderungen an den Testaufbau sind, über ein System zu verfügen, welches sowohl im Büro zum Testen neuer Applikationen und Komponenten als auch direkt am Motorenprüfstand zur Bedienung und Regelung verschiedener Hardware eingesetzt werden kann. Aus diesen beiden Hauptanforderungen entstand das im Folgenden beschriebene System. In Abbildung 4.1 ist der fertig verbaute Wagen zu sehen. Des Weiteren sind in Abbildung 4.2 die im hinteren Teil des Systems integrierten EtherCAT-I/O-Module dargestellt. Diese bilden zusammen mit dem AVL Frequenz-I/O die EtherCAT-Slaves, welche mit dem Kontroll-PC beziehungsweise dem Simulations-PC verbunden sind. Eine Gesamtübersicht der Module ist in Abbildung 4.3 zu sehen.

Das Kernstück des Testsystems bilden die beiden ident ausgestatteten, im oberen Bereich des Wagens verbauten PCs. Jeweils ausgestattet mit einer Xeon CPU E5345 (zwei CPUs mit je vier Cores, 2.33 GHz Taktfrequenz und 3.25 GByte RAM) und der Ethernet PCI-Karte "PRO/1000 GT Quad Port Server Adapter" von Intel, fungieren diese nicht nur als Laufzeitumgebung für Regelungs- und Simulationsapplikationen, sondern auch als EtherCAT-Master.

Das installierte EtherCAT-I/O-Modul der Firma Beckhoff besteht aus einem EtherCAT-Koppler, welcher den EtherCAT-Bus in einen LVDS-Bus übersetzt (siehe Abschnitt 3.4.1), sowie aus mehreren modularen I/O-Blöcken. Mithilfe dieses Moduls können die Ein- und Ausgangsdaten aus den jeweiligen Applikationen in analoge, digitale oder auch Frequenz-Signale umgesetzt werden.

Die in der Front des Wagens verbauten Panels sind, wie in Abbildung 4.3 zu sehen ist, mit den im Testsystem integrierten I/Os verbunden. Dies erlaubt einen flexiblen und unkomplizierten Zugriff auf die jeweiligen I/O-Module von außen. Mit vorkonfektionierten

Abbildung 4.1: Testsystem für den Einsatz im Büro als auch am Prüfstand

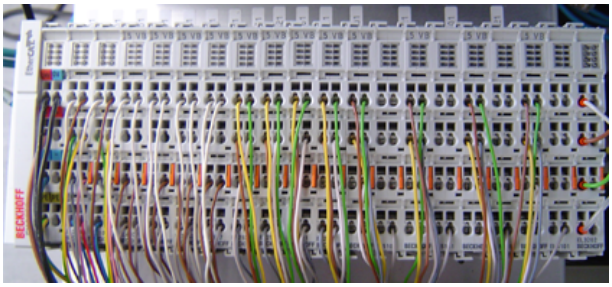


Abbildung 4.2: Im Wagen verbaute I/O-Leiste

Kabeln ist somit der Anschluss, an zum Beispiel die Geräte eines realen Prüfstandes, oder auch der "Kurzschluss" der integrierten Module, in kürzester Zeit durchzuführen. Für den universellen Einsatz, insbesondere an einem Prüfstand, sind im Kontroll- und Simulations-PC weiters CAN und Firewire-Karten installiert.

4.2 Testsystem - Software

4.2.1 Betriebssysteme und Laufzeitumgebung

Die Basis für alle in dieser Arbeit präsentierten Applikationen, bilden die Betriebssysteme. Sowohl auf dem Kontroll-PC als auch auf dem Simulations-PC, ist ein nicht Echtzeit-fähiges Betriebssystem (Windows XP) und parallel dazu ein Echtzeit-Betriebssystem (IN-time) installiert. Für eine detaillierte Ausführung sei an dieser Stelle auf den Abschnitt 2.1

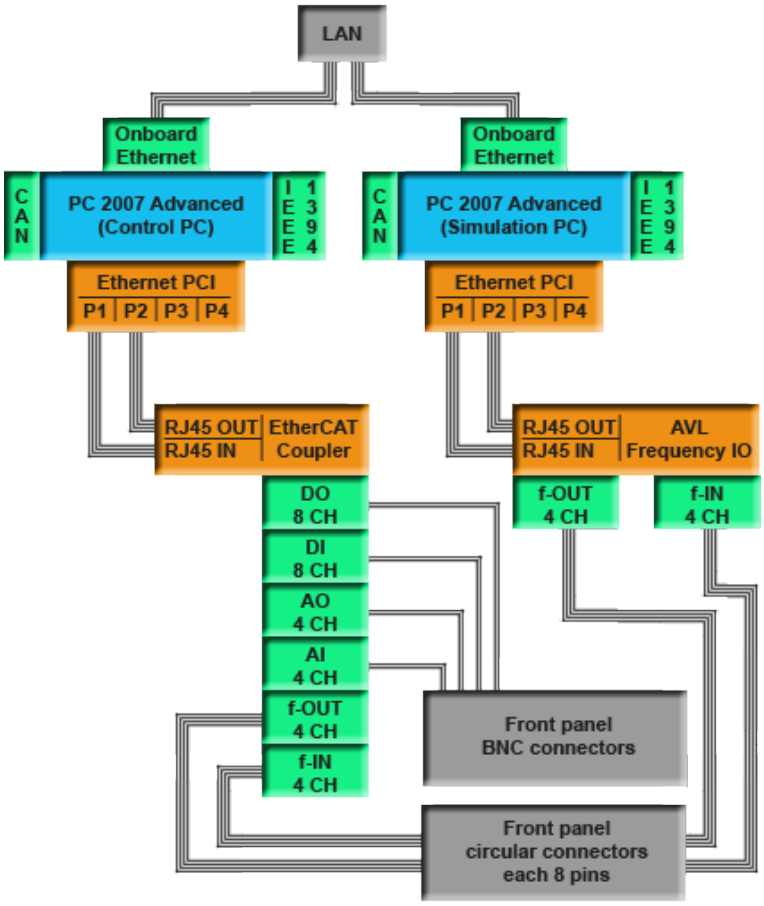


Abbildung 4.3: Hardware-Planung des Testsystems - Modulübersicht

verwiesen. Als Erweiterung von INtime dient das AVL Real-Time Environment (ARTE). Darüber kann sich die Software eines Automatisierungssystems befinden oder, wie im Fall des Simulations-PCs, eine einfache Umgebung zur Steuerung (zum Beispiel Starten und Stoppen) der Echtzeit-Applikationen.

AVL Real-Time Environment (ARTE)

Wie man in Abbildung 4.4 sehen kann, wurde ARTE als Schicht zwischen dem Echtzeit-Betriebssystem und dem Automatisierungssystem eingezogen. ARTE bietet alle Echtzeit-Dienste der darunter liegenden Schicht über ein standardisiertes Interface an. Weiters stellt es für diese Dienste angepasste Bibliotheken zur Verfügung, was die Entwicklung von Echtzeit-Software-Komponenten erheblich vereinfacht. Diese Bibliotheken ermöglichen es, für INtime- und Windows-Applikationen dieselbe Entwicklungsumgebung zu nutzen. Auch einige Prozesse des Echtzeit-Betriebssystems werden durch ARTE integriert. Einige der Hauptfunktionen sind das Tasksystem mit seinen 256 Prioritätsabstufungen, der schnelle Datenaustausch zwischen den Prozessen via gemeinsamen Speicher und der Zugriff auf Daten des Windows-Systems. Der letzte Punkt wird dadurch ermöglicht, dass ARTE sowohl auf die Win32 als auch auf die INtime System Calls Zugriff hat [Pri03].

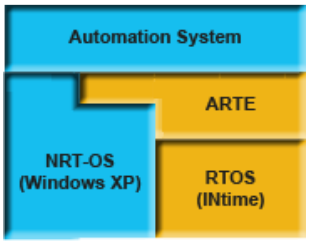


Abbildung 4.4: Software-Schichten des Echtzeit-Rechner-Systems

4.2.2 Simulationsmodelle

Alle in weiterer Folge gezeigten Modelle wurden mit dem Tool Matlab&Simulink implementiert. Die Portierung in Richtung Echtzeit-Applikation erfolgte mit Hilfe der zuvor erwähnten Bibliotheken von ARTE.

Modelle auf einem Rechner

Im ersten Schritt wurden nach Fertigstellung der Simulationsmodelle die einzelnen Komponenten, für den Betrieb auf einem Rechner, zu einem Modell zusammengefasst. Der Grund für diese Konfiguration liegt größtenteils in der Evaluierung der Modelle selbst, sowie deren Zusammenspiel. Im Folgenden sollen die vier Hauptkomponenten Fahrer (Driver), Regler (Controller), Motor (Engine) und Bremse (Dyno) genauer betrachtet werden. Sämtliche Abbildungen stellen den Aufbau in Simulink dar und sollen die Beschreibung der Funktionalität, sowie der Ein- und Ausgänge verdeutlichen. Tatsächlich liegen die Modelle in Form von Executables des Echtzeit-Betriebssystems vor.

Fahrer-Modell

Abbildung 3.5.4 zeigt den Aufbau des einfachen Fahrer-Modells. Als Eingänge werden ein zyklisch laufender Timer, sowie die aktuelle Drehzahl der Bremse benötigt. Als Ausgänge stehen die Sollwerte für die Drosselklappenstellung und Dynodrehzahl, der aktuelle Wert des Timers, sowie die beiden Steuerbit (Starter und Zündung) für den Motor zur Verfügung. Mittels Parameter können die Zeitpunkte der verschiedenen Ereignisse in Tabellen hinterlegt und während der Simulation durch den Cycle Timer als Trigger abgerufen werden. Die Standardfunktionalität beinhaltet folgenden Ablauf:

1. Start der Simulation - Cycle Timer wird mit 0 initialisiert.
2. Nach 30 Sekunden wird die Zündung eingeschaltet.
3. Nach 35 Sekunden wird für die Dauer von einer Sekunde der Starter betätigt.
4. Nach 40 Sekunden wird der erste Drehzahlsprung auf 1000 Umdrehungen pro Minute am Ausgang "Y_n_Dyno_Demand_Driver" durchgeführt.
5. Alle weiteren 20 Sekunden folgt ein weiterer Drehzahlsprung bis der Cycle Timer 100 Sekunden erreicht.
6. Cycle Timer wird auf 0 zurückgesetzt und die Zündung abgeschaltet.

7. Zyklus beginnt von vorne.

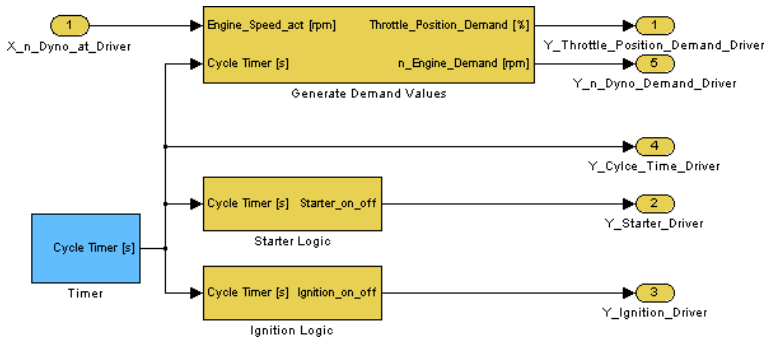


Abbildung 4.5: Einfaches Fahrer-Modell

Selbstverständlich gibt es weit aus komplexere und funktionsreichere Modelle für Fahrer. Für die Testszenarien rund um diese Arbeit wurde jedoch bewusst ein solch einfacher "Sollwertgeber" implementiert, um nicht durch unnötige Komplexität die Rekonstruierbarkeit der Sollwerte zu gefährden.

Regler-Modell

Das Regler-Modell (Abbildung 4.5) hat die Aufgabe den vom Fahrer erhaltenen Drehzahlwunsch in ein elektrisches Stellmoment für den Dyno umzuwandeln und dieses derart zu regeln, dass sich nach kurzer Zeit eine Gleichheit von Drehzahlwunsch und Istdrehzahl einstellt.

Da der Fokus dieser Arbeit nicht auf den Regelstrategien und bestmöglicher, regelungstechnischer Auslegung liegt, wurde hier ein einfacher PID-Regler verwendet. Die mathematische Beziehung von Eingangsgrößen zu Ausgangsgröße ist in den Gleichungen **??** und **??** ersichtlich.

$$Regelabweichung_{Dyno} = e_{Dyno} = (Solldrehzahl_{Dyno} - Istdrehzahl_{Dyno}) \tag{4.1}$$

$$Stellmoment_{Dyno} = K_P(e_{Dyno} + \frac{1}{T_I}\int e_{Dyno}dt + T_D\frac{d}{dt}e_{Dyno}) \tag{4.2}$$

Motor-Modell

Das Kernstück des virtuellen Motors ist ein 3D-Mapping aus dem sich, mit den Eingangsgrößen Drehzahl und Drosselklappenstellung, ein Drehmoment als Ausgangsgröße einstellt. Dieses Moment repräsentiert das innere Drehmoment des Motors, welches über ein PT1-Glied verzögert wird. Weiters wird es mit einem Reibmoment - dargestellt durch ein 2D-Mapping mit der Drehzahl als Eingang und dem Reibmoment als Ausgang - und einem Startermoment überlagert. Die Summe dieser drei Momente wirkt auf die darauffolgende Motorträgheit. Durch diese wird mittels des einfachen Zusammenhangs aus Gleichung **??** die aktuelle Drehzahl des Motors berechnet.

$$Motordrehzahl = \int \frac{\sum Momente}{Motorträgheit}dt \tag{4.3}$$

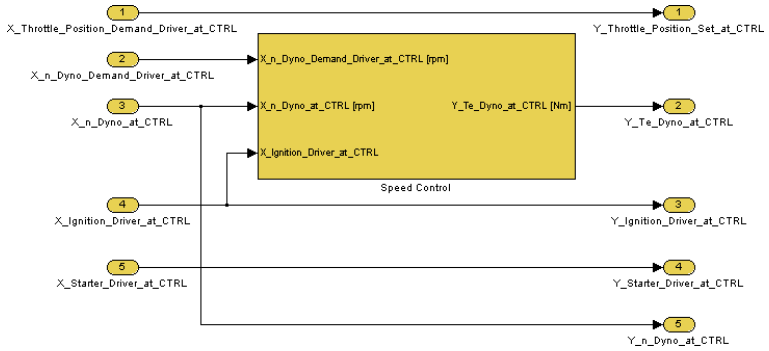


Abbildung 4.6: Regler-Modell

Der in Abbildung 4.6 dargestellte Eingang "X_T_Dyno_at_Engine" dient als Kopplung zur Bremse. Dieses Moment wird von der Summe der Motormomente subtrahiert und stellt dadurch die Last des Motors dar.

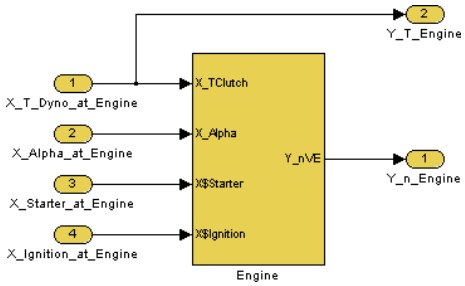


Abbildung 4.7: Motor-Modell

Dyno-Modell

Die Eingangsgrößen des Bremsen-Modells aus Abbildung 4.7 sind

- Motordrehzahl,
- elektrisches Stellmoment,
- Parameter zum (Ent-)koppeln der Bremse.

Die Ausgangsgrößen sind

- aktuelle Bremsendrehzahl und
- aktuelles Wellenmoment.

Die Implementation dieses Modells besteht im Wesentlichen aus den drei Teilen Trägheit, Reibmoment und Welle (Feder-Dämpfer Einheit). Die Trägheit fließt beim Dyno ähnlich wie beim Motor-Modell ein (siehe Gleichung **??**). Das heißt, die Summe der Momente wird

hier aus elektrischem Stellmoment, Reibmoment und aktuellem Wellenmoment gebildet. Die Welle wird durch den mathematischen Zusammenhang aus Gleichung 4.1 nachgebildet.

$$Wellenmoment = \int (Motordrehzahl - Dynodrehzahl)dt * Federkonstante$$
$$+ (Motordrehzahl - Dynodrehzahl) * Dämpferkonstante \quad (4.4)$$

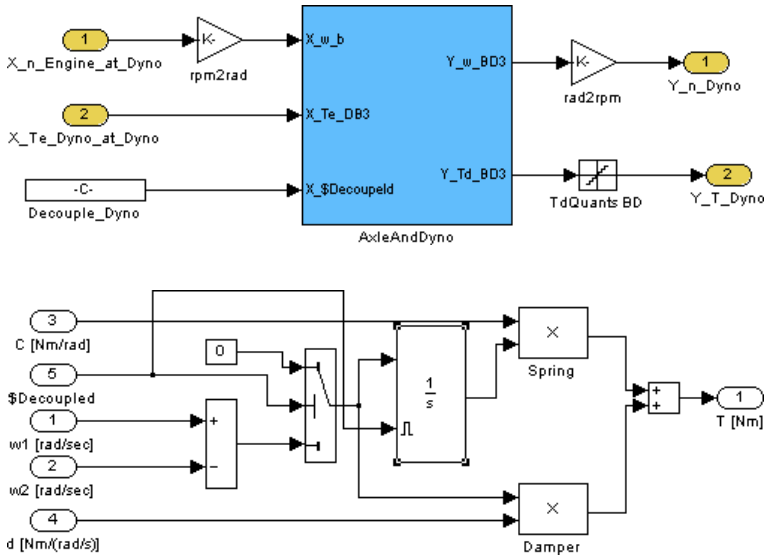


Abbildung 4.8: Dyno und Wellen-Modell

Verbindungsmodelle

Zusätzlich zu den vier oben angeführten Modellen, wurden Verbindungsmodelle mit parametrierbaren Signal-Verzögerungszeiten implementiert. Die Möglichkeit einer einstellbaren Signalverzögerung ist jedoch nicht der einzige Grund für diese Modelle. Wie man am Verbindungsmodell zwischen Motor und Bremse sieht (Abbildung 4.11), gibt es keine Verzögerung, da die beiden Komponenten in der Realität mechanisch gekoppelt sind. Der Grund dennoch ein Zwischenmodell einzuziehen, liegt in der vereinbarten Namenskonvention für Ein- und Ausgänge. Ohne Verbindungsmodell zwischen zum Beispiel Motor und Dyno müsste der Motordrehzahleingang an der Bremse ebenfalls mit "Y_n_Engine" benannt werden, was gegen diese Konvention verstoßen würde.

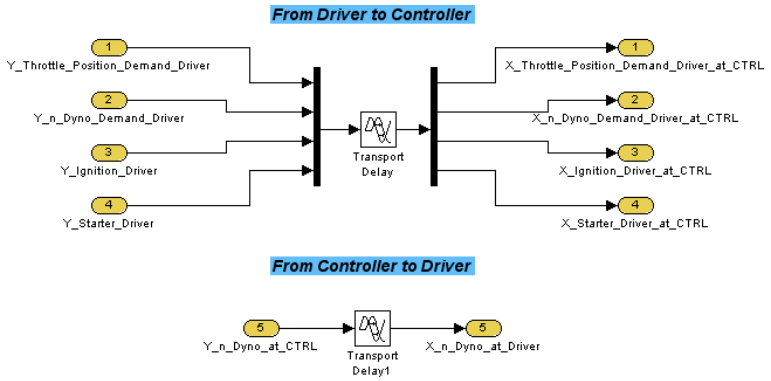


Abbildung 4.9: Verbindungsmodell zwischen Fahrer und Regler

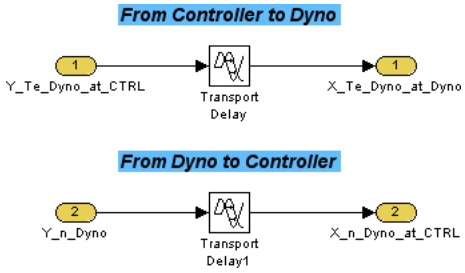


Abbildung 4.10: Verbindungsmodell zwischen Regler und Bremse

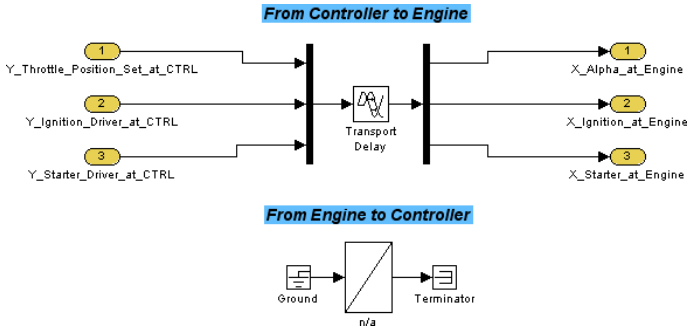


Abbildung 4.11: Verbindungsmodell zwischen Regler und Motor

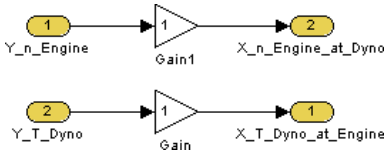


Abbildung 4.12: Verbindungsmodell zwischen Motor und Bremse

4.3 Einsatz- und Testszenarien

Die im Folgenden beschriebenen Szenarien stellen natürlich nur einen Bruchteil der Testmöglichkeiten dar, welche mit dem aufgebauten System möglich sind. Sie sollen jedoch einen Überblick über die möglichen Kategorien der Tests geben.

4.3.1 Software Tests

Tests der Automatisierungssoftware beziehungsweise deren Komponenten könnten ebenfalls mit der unter Abschnitt 4.2.2 beschriebenen Konfiguration "Modelle auf einem Rechner" durchgeführt werden. Dabei sind die zu überprüfende Software sowie die Prüfstandsimulation, also die Testumgebung, auf demselben Rechner integriert. Der Datenaustausch findet mit Hilfe der von ARTE zur Verfügung gestellten Funktionalität des gemeinsamen Speichers statt. Diese Variante wird jedoch nur selten verwendet, da zum Beispiel Aussagen über die Echtzeit-Betriebssystem-Last bei parallel laufenden, zu testenden Komponenten und Simulationsumgebung erschwert werden.

Virtueller Prüfstand - Modelle auf zwei Rechner verteilt

Ziel dieser Konfiguration ist es, einen Rechnerknoten mit dem zu testenden Automatisierungssystem und nötigem I/O auszustatten und dem zweiten Knoten die Aufgabe der Simulation eines realen Motorenprüfstandes zu übergeben. Die Kommunikation zwischen den beiden findet ausschließlich über das zur Verfügung stehende I/O statt, wobei der Automatisierungsknoten keinen Unterschied zwischen realem und simuliertem Prüfstand erkennen soll. Die Modelle aus Abschnitt 4.2.2 werden hier selbstverständlich weiter verwendet. Lediglich die Zusammenschaltung ändert sich dahingehend, dass die Modelle für Fahrer und Regler entfallen und durch das zu testende Automatisierungssystem ersetzt werden. Die Modelle für Motor und Dyno, sowie deren Verbindung, werden zu einem Modell zusammengefasst und als ausführbare Echtzeit-Applikation in die Laufzeitumgebung des Simulations-PCs geladen. Die zuvor verwendeten Verbindungsmodelle zwischen Regler und Motor beziehungsweise zwischen Regler und Dyno werden durch reale physikalische Verbindungen ersetzt. Die minimale Anforderung bezüglich des Signalaustausches ist in Tabelle 4.1 dargestellt. Der Typ des jeweiligen Signals, wurde anhand jenes Typen des tatsächlichen Sensors beziehungsweise Aktuators an einem realen Prüfstand festgelegt. Ein Überblick des Zusammenschlusses sowie der Kommunikation zwischen den Komponenten des virtuellen Prüfstandes ist in Abbildung 4.12 zu sehen.

Signalbeschreibung	*Signaltyp*	*Sender*	*Empfänger*
Regelung on/off	Digital	Kontroll-Rechner	Dyno
Zündung on/off	Digital	Kontroll-Rechner	Motor
Starter on/off	Digital	Kontroll-Rechner	Motor
Dyno Störung	Digital	Dyno	Kontroll-Rechner
Stellgröße (Drehmoment)	Frequenz	Kontroll-Rechner	Dyno
Stellgröße (Drosselklappe)	Analog	Kontroll-Rechner	Motor
Aktuelle Dynodrehzahl	Frequenz	Dyno	Kontroll-Rechner
Aktuelles Wellenmoment	Frequenz	Dyno	Kontroll-Rechner
Aktuelle Motordrehzahl	Frequenz	Motor	Kontroll-Rechner

Tabelle 4.1: Signalaustausch zwischen Automatisierungsknoten und (simuliertem) Prüfstand

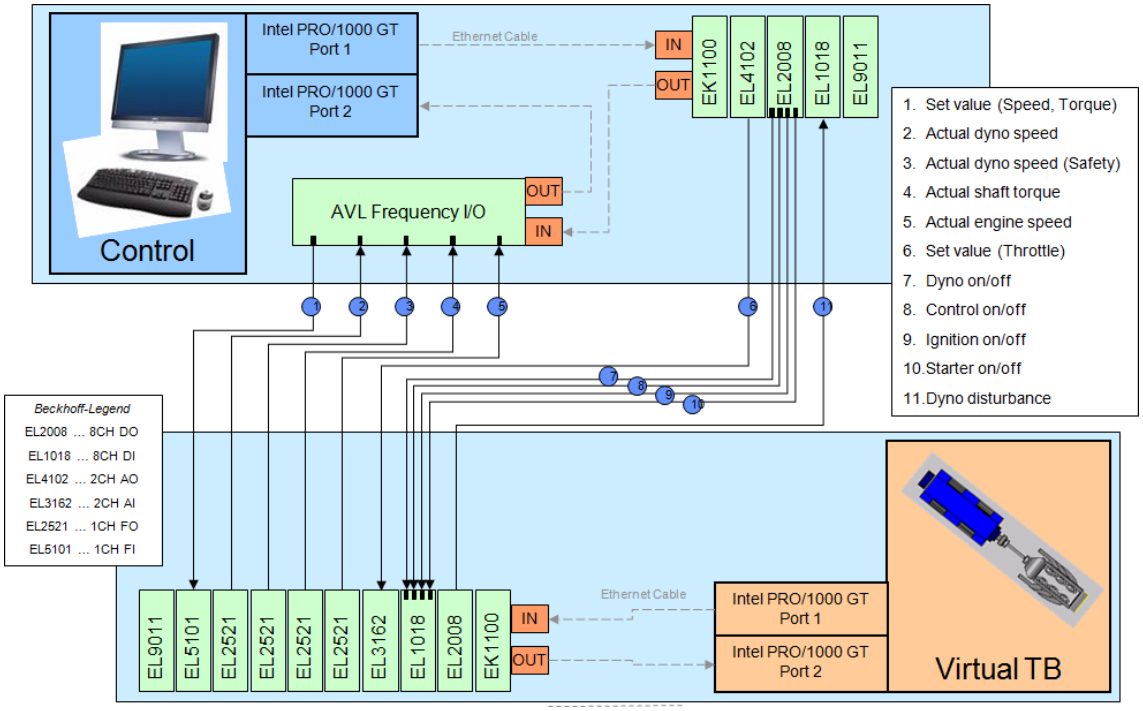


Abbildung 4.13: Signalaustausch zwischen Kontroll-Rechner und (simuliertem) Prüfstand

4.3.2 Tests am realen Prüfstand

Das in dieser Arbeit beschriebene System kann nicht nur zur Verifikation von neuen Hard- und Softwarekomponenten genutzt werden. Es besteht auch die Möglichkeit, damit einen realen Prüfstand zu steuern, was im Folgenden anhand eines Motorenprüfstandes gezeigt wird. Der Aufbau ähnelt sehr der zuvor beschriebenen Konfiguration des virtuellen Prüfstandes und ist überblicksmäßig in Abbildung 4.13 dargestellt. Einer der großen Vorteile des Systems ist hier schön zu erkennen. Durch die, wie in Abschnitt 4.1 gezeigt, auf den Panels aufgelegten I/O-Signale kann ohne viel Zeitaufwand zwischen simuliertem und realem Prüf-

stand gewechselt werden. Die Verkabelung wird einfach an die jeweiligen Prüfstands-I/Os umgelegt. Bei einer "vorausschauenden" Parametrierung der Komponenten am Automatisierungssystem, zum Beispiel der Drehzahl-Reglereinstellungen, während des Simulationsbetriebes, muss außer der Verkabelung nichts geändert werden und ein sofortiger Betrieb des realen Prüfstandes ist möglich. Natürlich geht es hierbei auch meist um einen Software-Test im erweiterten Sinn. Mit dieser Konfiguration kann gezeigt werden, dass die durch den virtuellen Prüfstand evaluierte Automatisierungssoftware auch am realen Prüfstand korrekt arbeitet.

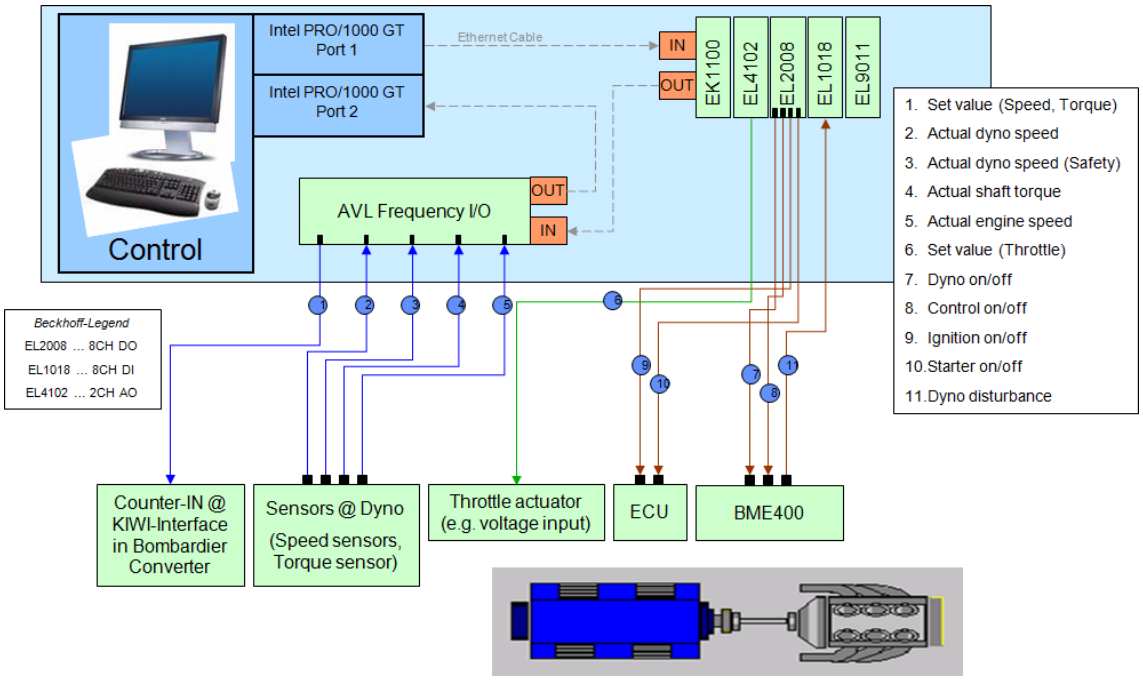


Abbildung 4.14: Signalaustausch zwischen Kontroll-Rechner und realem Prüfstand

4.3.3 Hardware-Komponenten Tests

Beispielhaft werden hier die Performance-Tests rund um das im Haus entwickelte Frequenz-I/O gezeigt. Da es sich dabei um einen EtherCAT-Slave handelt, sind nicht nur die Zeiten für die Umsetzung der Werte von den Eingängen beziehungsweise auf die Ausgänge im Gerät selbst interessant, sondern auch die benötigte Kommunikationszeit zwischen EtherCAT-Master - also dem Kontroll-PC - und dem Slave.

Closed Loop Test

Dieser Test soll das Zeitverhalten des Systems im Anwendungsfall "schnelle Regelung" darstellen. Der Aufbau ist somit an jene Konfiguration im Abschnitt 4.3.2 angelehnt. Der hauptsächliche Unterschied besteht darin, dass das zu messende Signal direkt auf einen Eingang am Frequenz-I/O zurückgekoppelt ist und aus diesem Grund eine meist große, aus dem mechanischen Aufbau entstehende Verzögerungszeit nicht ins Ergebnis mit einfließt.

Des Weiteren werden auch nur zwei Frequenzausgänge und ein Frequenzeingang verwendet, wobei der erste Frequenzausgang lediglich als Signalgenerator fungiert und in die Messung nicht mit einbezogen wird (siehe Abbildung 3.4.4). Die gemessene und am Oszilloskop dargestellte Zeit (siehe Abbildung 4.14) repräsentiert somit die Gesamtzeit der Schritte

1. AVL Frequency I/O: Lesen des am Frequenzeingang anliegenden Wertes
2. AVL Frequency I/O: Schreiben des am Frequenzeingang anliegenden Wertes auf den Bus
3. Control PC: Lesen des am Bus liegenden Wertes des Frequenzeingangs
4. Control PC: Umrechnung der Frequenzrohdaten in den tatsächlichen Wert
5. Control PC: Schreiben des auf dem zweiten Frequenzausgang auszugebenden Wertes auf den Bus
6. AVL Frequency I/O: Lesen des auszugebenden Wertes vom Bus
7. AVL Frequency I/O: Ausgabe des Wertes am zweiten Frequenzausgang

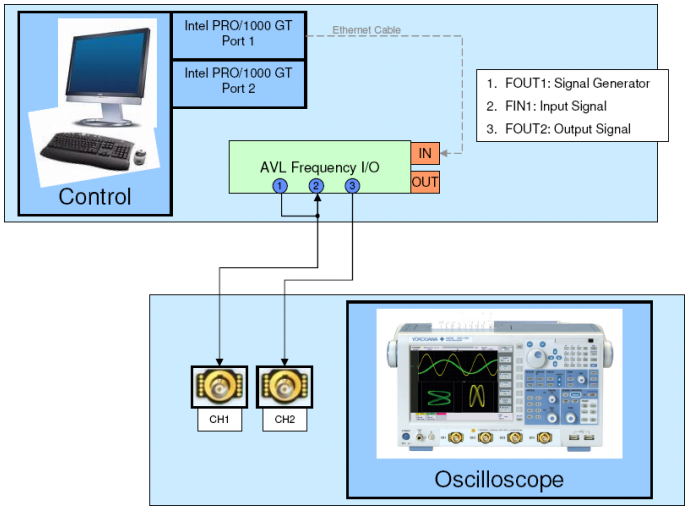


Abbildung 4.15: Performance-Messung Closed Loop Test

Um eine Verzerrung des Ergebnisses aufgrund von Streuung der Verzögerungszeiten zu verhindern wurden zwei Testläufe mit je drei Messungen durchgeführt und als repräsentativer Wert der Durchschnitt gewählt. In der Tabelle 4.2 sind die Ergebnisse der Messungen aufgelistet. Aus diesen folgt eine gemittelte Signallaufzeit von 407 µs.

Frequency I/O

Nachdem die benötigte Signallaufzeit des Closed Loop Tests ermittelt worden ist, sollen auch die einzelnen Komponenten genauer betrachtet werden. Mittels dem ET2000[1]

[1] Industrial-Ethernet-Multichannel-Probe Messgerät der Firma Beckhoff

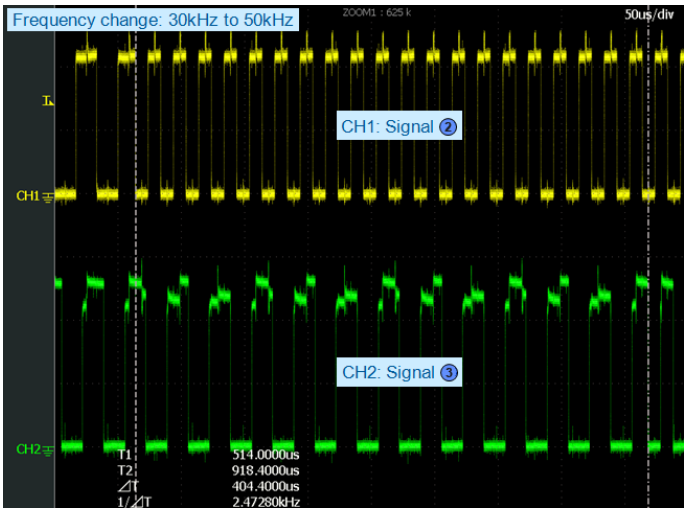


Abbildung 4.16: Darstellung der Verzögerungszeit am Oszilloskop

Testlauf #	*Frequenzsprung [kHz]*	*Systemfrequenz [kHz]*	*Verzögerung [µs]*
Messung 1 #1	30 → 50	10	403
Messung 1 #2	50 → 30	10	408
Messung 1 #3	30 → 50	10	390
Messung 2 #1	50 → 30	10	402
Messung 2 #2	30 → 50	10	410
Messung 2 #3	50 → 30	10	430

Tabelle 4.2: Verzögerungszeiten: Closed Loop Test

und der frei verfügbaren Software Wireshark, wird die Zeitspanne zwischen dem Anlegen eines Frequenz-Ausgangsignals und dem Eintreffen des physikalisch rückgekoppelten Frequenz-Eingangsignals am EtherCAT-Bus ermittelt. Das Automatisierungssystem läuft mit 10 kHz und triggert den Master somit alle 100 µs an. Es wird wieder ein Sprung von 30 kHz auf 50 kHz am Frequenzausgang und zurück im 100 Hz-Takt durchgeführt. Im Prozess Image des EtherCAT-Masters werden diese beiden Zahlen als 0x0060EA46 beziehungsweise 0x00504347 dargestellt. Wie man sieht, handelt es sich um 32 Bit Werte, welche sich aus einem 16 Bit Zeitstempel und einem ebenfalls 16 Bit langen Wert eines Flankenzählers zusammensetzen. Der mittels Wireshark aufgezeichnete Roh-Wert des Frequenz-Eingangs muss also, mittels dem in Gleichung **??** beschriebenen Zusammenhang, in den tatsächlichen Frequenz-Wert umgerechnet werden. Der Messaufbau ist in Abbildung 4.15 dargestellt. Die gemessene Zeit ergibt sich aus den einzelnen Schritten

1. Lesen des auszugebenden Wertes vom Bus
2. Ausgabe der Frequenz auf den ersten Frequenzausgang
3. Einlesen des physikalisch rückgekoppelten Wertes vom ersten Frequenzeingang
4. Zurückschreiben des Wertes am Frequenzeingang auf den Bus

$$f_{FIN\ aktuell} = \frac{\Delta_{Flanken} * f_{FPGA}}{\Delta_{Flanken\ Zeitstempel}} \tag{4.5}$$

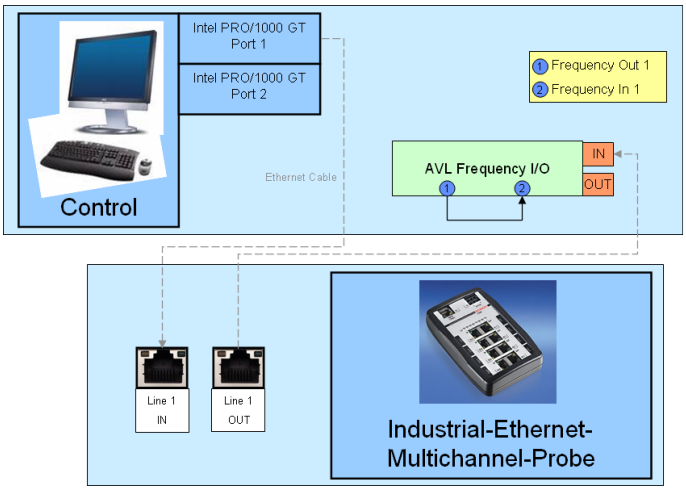


Abbildung 4.17: Performance-Messung am Frequency I/O

Um eine Verzerrung des Ergebnisses aufgrund von Streuung der Verzögerungszeiten zu eliminieren wurden zwei Testläufe mit je drei Messungen durchgeführt und als repräsentativer Wert der Durchschnitt gewählt. In der Tabelle 4.3 sind die Zeiten der einzelnen Messungen dargestellt. Als Mittelwert daraus ergibt sich eine Zeit von 284 µs. Diese Zeit repräsentiert die reine Umsetzungszeit für einen Wert am Frequenz-Ausgang auf den Wert des kurzgeschlossenen Frequenz-Eingangs der getesteten Hardware-Komponente. Zieht man diesen Durchschnittswert vom vorher gemessenen Wert der Closed Loop Messung ab, bleiben in etwas 120 µs für die Verarbeitung der Daten im Master sowie für die reine Kommunikation zwischen Master und Slave übrig.

Testlauf #	*Frequenzsprung [kHz]*	*Systemfrequenz [kHz]*	*Verzögerung [µs]*
Messung 1 #1	30 → 50	10	308
Messung 1 #2	50 → 30	10	307
Messung 1 #3	30 → 50	10	256
Messung 2 #1	50 → 30	10	310
Messung 2 #2	30 → 50	10	216
Messung 2 #3	50 → 30	10	308

Tabelle 4.3: Verzögerungszeiten: AVL Frequency I/O

Kapitel 5

Schlussbemerkung und Ausblick

Die in Abschnitt 4.3.3 dargestellten Signallauf- und Verzögerungszeiten sind signifikant höher, als jene Abschätzungen der minimalen Zykluszeiten aus Abschnitt 3.5.6. Der Hauptgrund dafür, liegt in der "hohen" Zykluszeit des Echtzeitbetriebssystems. Zurzeit ist für diese als Standardwert 100 µs vorgesehen. Das bedeutet, der Echtzeit-Ethernet-Master wird alle 100 µs, also mit einer Frequenz von 10 kHz getriggert. Somit ist es vollkommen klar, dass mit der derzeit vorliegenden Implementierung keine Zykluszeiten unter 100 µs erreicht werden können. Vereinzelt wurden bereits Tests mit 50 µs Zykluszeit des Echtzeitbetriebssystems, welche in einer Beta-Version zur Verfügung stand, durchgeführt. Dies zeigte eine erhebliche Verbesserung in punkto Latenzzeit der Übertragung, aber auch eine nicht unerhebliche, zusätzliche Rechenlast für die betreffenden CPU.

Weiters ist anzumerken, dass die Slave-Implementierungen natürlich dazu im Stande sein müssen, mit diesen Anforderungen mitzuhalten. Wie in den Tests dargestellt, benötigt zum Beispiel das neue Frequenz-I/O beträchtlich mehr Zeit zum Umsetzen eines Wertes auf einen Ausgang, beziehungsweise für das Lesen von einem Eingang, als 100 µs. Zusätzlich war bei den Messungen eine Schwankung der Latenzzeit von bis zu 30 Prozent zu bemerken. Dies hat den einfachen Grund, dass zum Zeitpunkt der in dieser Arbeit präsentierten Messungen, noch keine Synchronisierung des Masters mit den Slaves stattfand. Im Großen und Ganzen, gibt es definitiv Optimierungspotenzial, sowohl bezüglich der Master- als auch der Slave-Implementierungen.

In der Zwischenzeit, ergaben sich weitere Einsatzszenarien für sowohl das modulare AS und seinen Komponenten als auch für das entstandene Testsystem. Zum Beispiel werden zurzeit neue Slaves entwickelt und getestet, welche direkt in den Umrichtern der Dynos eines Prüfstandes integriert sind. Somit wird eine direkte Kommunikation zwischen Kontroll-PC und Umrichter-Elektronik ermöglicht. Dies hat den wesentlichen Vorteil, dass die zuvor beschriebene, zum jetzigen Zeitpunkt noch relativ hohe Verzögerungszeit des I/Os, welches bis dato zur Kommunikation mit der Umrichter-Elektronik eingesetzt wurde, von vorn herein entfällt.

Auch aus Sicht der schnellen Regelung von hochdynamischen Motorenprüfständen bringen die neuen Systeme ebenfalls einen bedeutenden Mehrwert und das nicht nur in technischer, sondern auch aus wirtschaftlicher Sicht. Bisher war in diesem Bereich der Zukauf von Hard- und Software eines Drittanbieters notwendig, um die geforderte Performance zu erreichen. Die Kosten für derartige Erweiterungen können überschlagsmäßig im Bereich von mehreren tausend Euro, bis hin zu über einhundert tausend Euro pro Prüfstand beziffert

werden. Dies hat sich mit dem Ansatz des modularen AS und der darin enthaltenen Möglichkeit, eigene unter zum Beispiel Simulink aufgebaute Regler in Echtzeit zu betreiben, erübrigt.

Abschließend kann gesagt werden, dass das entstandene Testsystem seinen Zweck mehr als erfüllt hat und auch die damit evaluierten, neuen Komponenten stets den technologisch richtigen Weg in die Zukunft erkennen ließen. Bleibt nur zu hoffen, dass sich die zurzeit allgemein angespannte wirtschaftliche Situation rasch erholt und dadurch auch die Aktivitäten in den Entwicklungsbereichen wieder verstärkt werden können.

Anhang A

Abkürzungen und Markenzeichen

A.1 Verwendete Abkürzungen

API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
ARTE	AVL Real-Time Environment
AS	Automation System, Automatisierungssystem
ASIC	Application-specific integrated circuit
CAN	Controller Area Network
CBA	Component Based Automation (PROFINET)
CFI	Canonical Format Indicator
CIP	Common Industrial Protocol
CN	Controlled Node
CPU	Central processing unit
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
EMV	Elektro-magnetische Verträglichkeit
EPSG	Ethernet POWERLINK Standardization Group
ESD	End-of-Stream Delimiter
ETG	EtherCAT Technology Group
EtherCAT	Ethernet for Control and Automation Technology
FCS	Frame Check Sequence Field

FDDI	Fiber Distributed Data Interface
FLP	Fast Link Pulse
FMMU	Fieldbus-Memory-Management-Unit
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GSD	General Station Description
HAL	Hardware Abstraction Layers
HMI	Human Machine Interface
HTTP	Hypertext Transfer Protocol
ID	Identifier
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFG	Inter-frame gap
IP	Internet Protocol
IRT	Isochronous Real-Time
ISO	International Standardization Organisation
LAN	Local Area Network
LLC	Logical Link Control
LSB	Least significant bit
LVDS	Low-Voltage-Differential-Signal
LWL	Lichtwellenleiter
MAC	Media Access Control
MDC	Management Data Clock
MDI	Medium Dependent Interface
MDIO	Management Data I/O
MII	Media Independent Interface
MLT-3	Multilevel Treshold-3
MN	Managing Node

MSB Most significant bit

NLP Normal Link Pulse

NMT Netzwerk-Management

NRZI No Return to Zero/Inverted

NTX Windows NT extension

ODVA Open DeviceNet Vendor Association

OSI Basic Reference Model for Open Systems Interconnection

PC Personal Computer

PCS Physical Coding Sublayer

PDO Prozess- Daten-Objekte

PHY Physical Layer

PI Prozess-Image

PMA Physical Medium Attachment

PMD Physical Medium Dependent Sublayer

PMM Permanent Magnet Machines

PROFIBUS Process Field Bus

PROFINET Process Field Net

RAM Random-Access Memory

RFC Request for Comments

RT Real-Time

RTCS Real-Time Computer System

RTOS Real-Time Operating System

RTPS Real-Time Publish-Subscribe (ModBus)

RSL Real-time Shared Library

SCNM Slot Communication Network Management

SDO Service-Daten-Objekte

SERCOS Serial Realtime-Communication System

SFD Start-of-Frame Delimiter

SoC Start of Cycle

SSD Start-of-Stream Delimiter

TCP Transmission Control Protocol

UDP User Datagram Protocol

VLAN-TPID Virtual Local Area Network - Tag Protocol Identier

XML Extensible Markup Language

A.2 Eingetragene Marken

Alle eingetragenen Marken beziehungsweise Markenzeichen von in dieser Arbeit erwähnten Technologien, Firmen und Organisationen werden hiermit anerkannt. Um die Lesbarkeit zu vereinfachen wurde im Text auf die wiederholte Darstellung der betreffenden Namen als eingetragenes Markenzeichen verzichtet. Dies soll an dieser Stelle stellvertretend für den gesamten Text in alphabetischer Reihenfolge nachgeholt werden.

Marke	*Firma bzw. Organisation*	
Intel®	Intel®Corporation	`http://www.intel.com`
Intel®Xeon®	Intel®Corporation	`http://www.intel.com`
INtime®	TenAsys© Corporation	`http://www.tenasys.com`
Matlab®& Simulink®	The Mathworks™	`http://www.mathworks.com`
Windows©	Microsoft®	`http://www.microsoft.com`

Tabelle A.1: Verwendete eingetragene Marken

Literaturverzeichnis

[Bor08] Kai Borgeest. *Elektronik in der Fahrzeugtechnik.* Kraftfahrzeugtechnik. Friedr. Vieweg & Sohn Verlag, 2008.

[CiA10] CAN in Automation e.V. Online-Resource, April 2010. http://www.canopen.de/, eingesehen am 21. April 2010.

[CS09] Inc. Cisco Systems. Internetworking Technology Handbook, January 2009.

[Dan06] Wilfried Dankmeier. *Grundkurs Codierung.* Friedr. Vieweg & Sohn Verlag, 2006.

[EPS08] Ethernet POWERLINK Systemübersicht. Technical report, Ethernet POWERLINK Standardisation Group, 2008.

[ETG08] EtherCAT - Der Ethernet-Feldbus. Technical report, EtherCAT Technology Group, December 2008.

[Fel09] Max Felser. Das PROFINET Handbuch. Online-Resource, 2009. http://www.profinet.felser.ch/, eingesehen am 27.April 2010.

[Gru08] Werner Gruber. Automatic Distribution of Large Real-Time Task Sets for Multi-Core based Test-Bed Systems. Master's thesis, Institute for Technical Informatics, Graz University of Technology, 2008.

[J+07] Juergen Jasperneite et al. Limits of Increasing the Performance of Industrial Ethernet Protocols. In *12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 17–27, Patras, Greece, September 2007.

[K+08] Stephan Kirchmayer et al. Ethernet POWERLINK Communication Profile Specification. Technical report, 2008.

[Kop97] Hermann Kopetz. *Real-time systems - design principles for distributed embedded applications.* Kluwer, Boston, Mass., 1997.

[Kra08] Jens Onno Krah. Gut integriert - Motion Control mit FPGA und Echtzeit-Ethernet. *ethernet 2008*, pages 26–29, 2008.

[Leb08] Prof. Dr.-Ing. Klaus Lebert. Vorlesungsunterlagen Steuerungstechnik - WS 2008/2009, 2008.

[Luk07] Andrey Lukyanenko. On the Optimality and the Stability of Backoff Protocols. In *Next Generation Teletraffic and Wired/Wireless Advanced Networking*, Lecture Notes in Computer Science, pages 393–408, St. Petersburg, Russia, 2007.

[Old09] Universität Oldenburg - Signalkodierung. Online-Resource, 2009. http://einstein.informatik.uni-oldenburg.de/rechnernetze/signalko.htm, eingesehen am 05. Jänner 2009.

[PM08] Raimond Pigan and Mark Metter. *Automating with PROFINET*. Publicis Publishing, Erlangen, 2nd edition, 2008.

[Pri03] Dietmar Franz Prisching. *Performance Evaluation and Prediction in Complex Real-Time Automation Systems*. PhD thesis, Graz University of Technology, 2003.

[Pro09] PROFINET Systembeschreibung. Technical report, PROFIBUS Nutzerorganisation e.V., Karlsruhe, Germany, April 2009.

[Pry08] Gunnar Prytz. A performance analysis of EtherCAT and PROFINET IRT. In *13th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 408–415, Hamburg, Germany, September 2008.

[Rec08] Joerg Rech. *Ethernet*. Heise Zeitschriften Verlag, 2008.

[Rei10] Gerald Reiner. Wirtschaftsuniversität Wien - Production Management. Vorlesungsunterlagen, 2010. http://indi.wu-wien.ac.at/ reiner/pm-ps/begleitmaterial/Teil_09.pdf/.

[Rig05] Wolfgang Riggert. *Rechnernetze Grundlagen - Ethernet - Internet*. Fachbuchverlag Leipzig, 2005.

[Ros08] Martin Rostan. Industrial Ethernet Technologies: Overview. Presentation at ETG Industrial Ethernet Seminar Series, November 2008.

[SC05] Stanislav Shalunov and Richard Carlson. Detecting Duplex Mismatch on Ethernet. In *Passive and Active Network Measurement*, pages 135–148, Boston, MA, USA, 2005.

[Sch09] Prof. Dr.-Ing. Juergen Schwager. Informationsportal fuer Echtzeit-Ethernet in der Industrieautomation. Online-Resource, 2009. http://www.echtzeit-ethernet.de/, eingesehen am 26. Jänner 2010.

[Sha48] Claude E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, pages 379–423,623–656, October 1948.

[SW+06] Gerhard Schnell, Bernhard Wiedmann, et al. *Bussysteme in der Automatisierungs und Prozesstechnik*. Friedr. Vieweg & Sohn Verlag, 2006.

[Ten08] TenAsys Corporation, Beaverton, USA. *INtime 3.1 Software*, Jänner 2008.

[Ten09] TenAsys Corporation,, 2009. http://www.tenasys.com/, eingesehen am 10. September 2009.