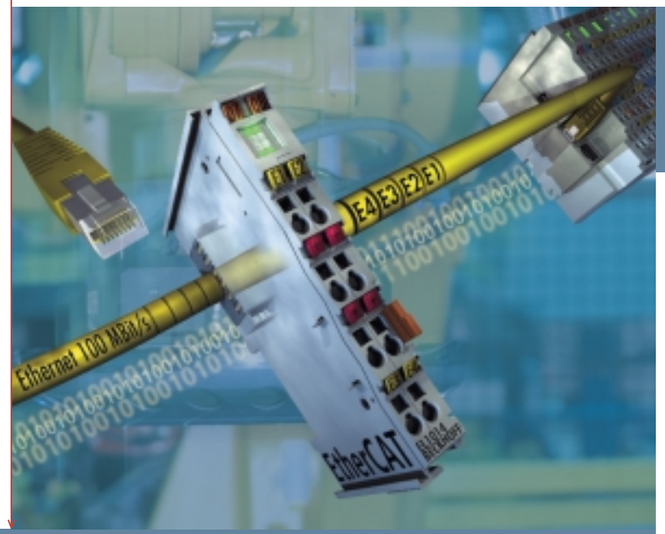
Die technische Antwort auf die Frage: „Warum EtherCAT?”

EtherCAT – Der Ethernet-Feldbus



In der weltweiten Automatisierungstechnik nimmt der Einsatz von Ethernet als physikalische Kommunikationsschicht rasant zu. Beginnend mit der Leitebene, über die Fabrikvernetzung bis hin zur Intersteuerungskommunikation ist Ethernet bereits etabliert und aufgrund der, aus der Office-Welt bekannten, hohen Komponenten-Stückzahlen vergleichsweise leistungsfähig und kostengünstig.

In der Feldebene dominieren weiterhin die in den 90er Jahren entwickelten Feldbusse. Mit unterschiedlichen Schwerpunkten haben sich die Feldbusse ihr jeweiliges Marktsegment geschaffen und mit maximalen Bruttoübertragungsraten von 0,5 – 16 MBaud sind sie auch weiterhin für viele Anwendungen ausreichend. Die, im Vergleich zur allgemeinen IT-Welt, verschwindend geringen Stückzahlen führen jedoch zu hohen Gesamtkosten – sowohl in der Steuerung als auch in den Feldgeräten und insbesondere in der Verkabelung. Die notwendige Flexibilität bezüglich der Signalgranularität wurde und wird durch modulare Feldgeräte erreicht, die einen unterlagerten Sub-Bus betreiben. Ein Sub-Bus erhöht die Komplexität des Gesamtsystems und hat signifikante Performance-Nachteile bezüglich Totzeiten und erreichbarer Zykluszeit.

Warum EtherCAT?

Während das Moore’sche Gesetz zumindest bei den PC-basierten Steuerungen in den letzten Jahren weiterhin Gültigkeit behalten hat und in Zukunft behalten wird (Verdoppelung der Leistungsfähigkeit ca. alle 2 Jahre), gab es bei den Feldbussen keine nennenswerten Weiterentwicklungen. Stattdessen wurde Ethernet mehrheitlich als kommende Ergänzung bzw. Ersatz für die „proprietäre“ Feldbustechnik sowohl von Feldbus-Organisationen als auch von „großen“ Automatisierungsfirmen ausgewählt, und entsprechende Standards wurden entworfen.
An dieser Stelle muss sich eine Firma wie Beckhoff fragen lassen: „Wieso noch ein weiterer Ethernet-Ansatz?“ Die kurze, aber prägnante Antwort lautet: „EtherCAT geht andere Wege und ist aktuell mit Abstand der leistungsfähigste und der am besten auf automatisierungstechnische Anforderungen zugeschnittene Ethernet-Ansatz!“ Eine fundierte, technisch begründete Antwort darauf folgt in diesem Artikel.

Ethernet macht auch Probleme

Ethernet hat im Vergleich zu den existierenden Feldbussen nicht nur Vorteile. Die für den Einsatz in der Automatisierungstechnik eher ungünstigen Eigenschaften von Ethernet müssen daher sehr genau betrachtet und möglichst optimal umgangen werden. Darauf beruhen auch die wesentlichen Unterschiede der verschiedenen Ansätze, Ethernet für die Automatisierungstechnik anzupassen:

- Hoher Overhead bei der Kommunikation mit Teilnehmern, die zwar häufig aber nur wenige Daten austauschen müssen.
- Hohe Anschaltkosten pro Teilnehmer (Trafo, Phy, Mac und notwendige Prozessor-Performance) verglichen mit klassischen Feldbusanschaltungen.
- Mangelnde Echtzeitfähigkeit, die bei genauerer Betrachtung aber weniger durch Ethernet als Übertragungsmedium verursacht wird sondern auf die ungünstigen Laufzeiten in den Software-Stacks zurückgeht.
- Ungünstige Topologie. Die inzwischen bei Ethernet übliche sternförmige Topologie ist bei der Anlagenverdrahtung eher ungünstig und führt entweder zu hohem Verkabelungsaufwand oder zu stark kaskadierten Kommunikationsbeziehungen.

Ethernet ist auch nicht gleich Ethernet. Neben den unterschiedlichen Übertragungsgeschwindigkeiten von 10, 100 und 1000 MBaud muss auch zwischen Halb- und Vollduplex-Kommunikation unterschieden werden. Halbduplex – also die Datenübertragung, die abwechselnd in die eine und in die andere Richtung geht – überträgt, im Vergleich zu Vollduplex, weniger als die Hälfte, da nach der Übertragung in die eine Richtung Pausenzeiten eingehalten werden müssen, um die Lauf- und Reaktionszeiten zu kompensieren. Bei Halbduplex-Übertragungen können Kollisionen auftreten, die zwar auf den untersten Ebenen abgefangen und durch Wiederholungen beseitigt werden, für eine deterministische Übertragung aber nicht akzeptabel sind. Daher müssen die Kollisionen auf höheren Ebenen vermieden werden, was aber wiederum nicht ohne weitere Verschlechterung der Nutzdatenrate möglich ist.

Ziele bei der Entwicklung von EtherCAT

PC-basierte Steuerungstechnik kann aufgrund der Leistungsfähigkeit des PC-Systems eine eher zentrale oder – genauer ausgedrückt – eine hierarchische Steuerungsphilosophie unterstützen, bei der alle auf einer Ebene relevanten Informationen gleichzeitig in einer Steuerung bekannt sind und miteinander kombiniert werden können. Dies erleichtert nicht nur die Konfiguration des Systems, sondern ermöglicht auch den Einsatz intelligenterer Steuerungsalgorithmen. Das Kommunikationssystem ist dabei „nur noch“ für den schnellen Transport der Prozessdaten in und aus der Steuerung verantwortlich. Ein Master-Slave-Kommunikationssystem ist für diesen Zweck die beste Wahl. Bei der Entwicklung von EtherCAT standen die folgenden Hauptziele im Vordergrund:

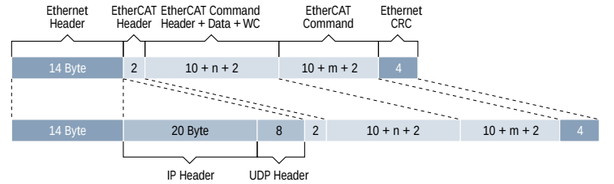
- Breite Einsetzbarkeit. Als EtherCAT-Master soll jede Steuerung mit einem handelsüblichen Ethernet-Controller eingesetzt werden können. Angefangen von einem kleinen 16 Bit µC bis hin zu PC-Systemen mit 3 GHz Prozessoren soll jeder Rechner ohne spezielle Anschaltung zur EtherCAT-Steuerung werden können.
- Vollständige Konformität zum Ethernet-Standard. EtherCAT soll mit anderen Ethernet-Geräten und -Protokollen am selben Bus koexistieren. Standard-Struktur-Komponenten wie Ethernet-Switche sollen für EtherCAT einsetzbar sein.
- Kleinstmögliche Teilnehmer-Granularität ohne unterlagerten Sub-Bus. Als EtherCAT-Slave sollen sowohl komplexere Knoten als auch 2-Bit-I/Os wirtschaftlich eingesetzt werden können.
- Höchstmögliche Effizienz. Die Bandbreite von Ethernet soll möglichst vollständig für Nutzdatentransfers zur Verfügung stehen.
- Kurze Zykluszeiten. Mögliche Zykluszeiten deutlich unter 100 µs sollen neue Anwendungsgebiete erschließen, wie z.B. das Schließen unterster Regelkreise in der Antriebstechnik.
- Höchste Deterministik, auch ohne die Basis absoluter Telegramm-Sendegenauigkeit.

EtherCAT-Funktionsprinzip

Aus Ethernet-Sicht ist ein EtherCAT-Bus nichts anderes als ein einzelner großer Ethernet-Teilnehmer. Dieser „Teilnehmer“ empfängt und sendet Ethernet-Telegramme.
Innerhalb des „Teilnehmers“ befindet sich aber kein Ethernet-Controller mit nachgeschaltetem µ-Prozessor, sondern eine Vielzahl von EtherCAT-Slaves. Diese verarbeiten die einlaufenden Telegramme im Durchfluss und nehmen die für sie bestimmten Nutzdaten heraus bzw. blenden sie ein und leiten das Telegramm an den nächsten EtherCAT-Slave weiter. Der letzte EtherCAT-Slave schickt das bereits vollständig verarbeitete Telegramm zurück, so dass es vom ersten Slave – quasi als Antworttelegramm – zur Steuerung zurückgeschickt wird. Es wird dabei ausgenutzt, dass Ethernet eine getrennte Übertragung in Hin- und Rückrichtung (Tx- und Rx-Leitungen) besitzt und im Vollduplex-Modus arbeitet. Natürlich kann, wie bei jedem anderen Ethernet-Teilnehmer auch, eine direkte Kommunikation ohne Switch mit einem „gedrehten“ Ethernet-Kabel aufgebaut werden, wodurch ein reines EtherCAT-System entsteht

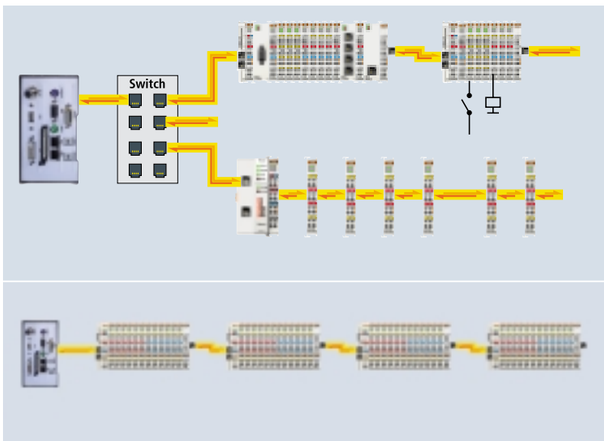
Telegrammverarbeitung

Die Verarbeitung der Telegramme findet im Durchlauf statt. Während die Telegramme, nur um wenige Bits verzögert, bereits weitergeschickt werden, erkennt der Slave für sich bestimmte Kommandos und führt sie entsprechend aus. Die Verarbeitung findet in Hardware statt und ist daher unabhängig von den Reaktionszeiten eventuell angeschlossener µ-Prozessoren. Jeder Teilnehmer besitzt dabei einen adressierbaren Speicherbereich von 64 kByte innerhalb dessen gelesen, geschrieben oder gleichzeitig geschrieben und gelesen werden kann. Innerhalb eines Ethernet-Telegramms können mehrere EtherCAT-Kommandos eingebettet werden, die jeweils individuelle Teilnehmer und/oder Speicherbereiche ansprechen. Die EtherCAT-Kommandos werden im Datenbereich eines Ethernet-Telegramms transportiert und können entweder über einen speziellen Ether-Type oder per UDP/IP kodiert sein.



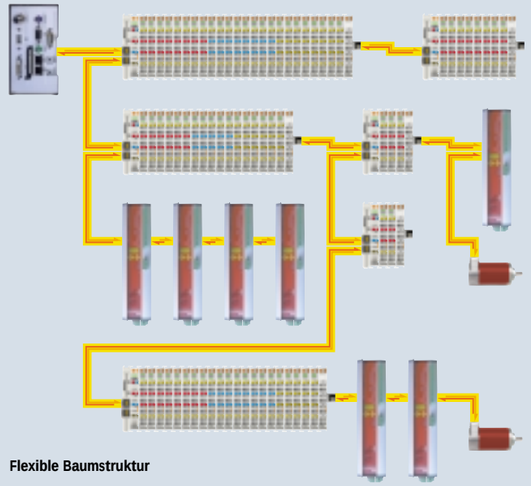
EtherCAT-Telegrammaufbau (ohne und mit UDP)

Die erste Variante mit speziellem Ether-Type ist zwar auf ein Ethernet-Subnetz begrenzt, d.h. entsprechende Telegramme werden von Routern nicht weitergeleitet, für Steuerungsaufgaben stellt dies in der Regel jedoch keine Einschränkung dar. Als Adressierung wird die Ethernet-MAC-Adresse des ersten Teilnehmers genutzt. Hierbei wird ein spezieller erster EtherCAT-Teilnehmer benötigt, der im Falle einer direkten Ansteuerung ohne Switch entfallen kann.
Die zweite Variante per UDP/IP erzeugt einen geringfügig größeren Overhead (IP- und UDP-Header), ermöglicht aber, bei weniger zeitkritischen Anwendungen, einerseits das normale IP-Routing zu nutzen und andererseits die häufig bereits vor-



Übertragungsschicht	Kabel	Max. Länge	Kosten (Anschaltung/lfd. Meter)	Konfektionierbarkeit	Verzögerung (pro Anschaltung in µs)
Ethernet (100 Base TX)	CAT 5 (Kupfer)	100 m	o / ++ (+[1])	+	ca. 1
Ethernet (100 Base FX)	LWL (Glasfaser)	100 km	- / --	-	ca. 1
E-Bus (Kupfer)	CAT 5 (Kupfer)	10 m	++ / ++ (+[1])	+	0
E-Bus (LWL, in Planung)	LWL (Plastikfaser)	100 m	+ / +	++	0

Eigenschaften der verschiedenen physikalischen Schichten
([1] industrielle Kabelausführung d.h. schleppkettentauglich, ölfest)



Flexible Baumstruktur

handenen TCP/IP-Stacks auf der Masterseite zu verwenden. Jedes EtherCAT-Kommando besteht aus einem EtherCAT-Header, dem Datenbereich und einem anschließenden Zählerfeld (Working-Counter), das von allen EtherCAT-Teilnehmern inkrementiert wird, die durch das EtherCAT-Kommando angesprochen wurden und entsprechende Daten ausgetauscht haben.

Fieldbus-Memory-Management-Unit (FMMU)

Mit Hilfe des beschriebenen Telegrammaufbaus können bereits mehrere EtherCAT-Teilnehmer durch ein einzelnes Ethernet-Telegramm mit mehreren EtherCAT-Kommandos angesprochen werden, was zu einer deutlichen Verbesserung der Nutzdatenrate führt. Für den Fall einer 2-Bit-Eingangsklemme, die eben genau 2 Bit Nutzdaten besitzt, ist der Overhead eines einzelnen EtherCAT-Kommandos aber weiterhin deutlich zu groß.

Die Fieldbus-Memory-Management-Unit beseitigt das Problem und ermöglicht eine nahezu 100 prozentige Nutzdatenrate – auch bei Teilnehmern, die, wie beschrieben, nur 2 Bit Nutzdaten aufweisen. Ähnlich wie eine Memory-Management-Unit (MMU) in modernen Prozessoren, übersetzt die FMMU eine logische Adresse anhand einer internen Tabelle in eine physikalische. Die FMMU ist im EtherCAT-Slave-Asic integriert und ermöglicht für jeden Teilnehmer individuell ein entsprechendes Adressmapping.

Im Unterschied zu prozessorinternen MMUs, die komplette Speicherseiten (im Bereich von 4 kByte) mappen, unterstützt die FMMU auch bitweises mappen. Dadurch können die zwei Bit der Eingangsklemme individuell in einen beliebigen Bereich eines logischen Adressraumes eingeblendet werden. Wird nun ein EtherCAT-Kommando verschickt das, anstatt einen speziellen EtherCAT-Teilnehmer anzusprechen, einen bestimmten logischen Speicherbereich liest oder schreibt, blendet die 2-Bit-Eingangsklemme ihre Daten an der richtigen Stelle in den Datenbereich ein. Alle anderen Klemmen, die ebenfalls einen Adressmatch in ihrer eigenen FMMU feststellen, blenden ebenfalls ihre Daten ein, so dass mit einem einzelnen EtherCAT-Kommando viele Teilnehmer gleichzeitig angesprochen werden. Da die FMMU in jedem Teilnehmer vorhanden ist und individuell konfiguriert wird, kann der Master in der Initialisierungsphase bereits komplette Prozessabbilder zusammenstellen und anschließend mit einem einzelnen EtherCAT-Kommando austauschen. Es wird kein zusätzliches Mapping mehr im Master benötigt, so dass die Prozessdaten direkt den unterschiedlichen Steuerungstasks (SPS, NC etc.) zugeordnet werden können. Jede Task kann ihr eigenes Prozessabbild erstellen und in ihrem eigenen zeitlichen Rahmen austauschen. Die physikalische Reihenfolge der EtherCAT-Teilnehmer ist dabei völlig unabhängig und spielt nur in der allerersten Initialisierungsphase eine Rolle.

Serielle Backplane für Automatisierungssysteme

Der logische Adressraum für die FMMUs beträgt 4 GByte. Aus Sicht des Masters kann ein EtherCAT-System daher als großer verteilter Speicher (Distributed-Memory) angesehen werden, in den wahlfrei geschrieben und gelesen werden kann. EtherCAT ist daher eine Art serielle Backplane für Automatisierungssysteme, die für große aber auch ganz kleine Automatisierungsgeräte die Anbindung an verteilte Prozessdaten ermöglicht. Über einen Standard-Ethernet-Controller und Standard-Ethernetkabel (CAT 5) können quasi beliebig viele und beliebig verteilte I/O-Kanäle an Automatisierungsgeräte angeschlossen werden, auf die mit hoher Bandbreite, geringster Verzögerung und nahezu optimaler Nutzdatenrate zugegriffen werden kann.

Physikalische Schicht

Für Ethernet existiert eine Vielzahl unterschiedlicher physikalischer Medien; angefangen vom in die Jahre gekommenem „Yellow Cable“ bis zu Hochgeschwindigkeits-Glasfaser-Strecken. Bei dem von EtherCAT verwendeten 100 MBaud Ethernet gibt es – neben dem klassischen CAT-5-Kupferkabel (100 Base TX) – auch Lichtwellenleiter-Übertragungsschichten (100 Base FX). Da EtherCAT vollständig kompatibel zu Ethernet ist, können alle entsprechenden Übertragungsschichten genutzt werden. Bei EtherCAT erfolgen häufig Übertragungen auf sehr kurzer Strecke, z. B. zwischen zwei EtherCAT-Klemmen im selben Klemmenblock, so dass zusätzlich eine weitere Übertragungsschicht entwickelt wurde – der E-Bus. Der E-Bus basiert auf einer LVDS-Übertragung (Low-Voltage-Differential-Signal, IEEE Standard P1596.3-1995) und ist neben der Klemmenkommunikation auch für kostengünstige „Fernübertragungen“ bis ca. 10 m geeignet.

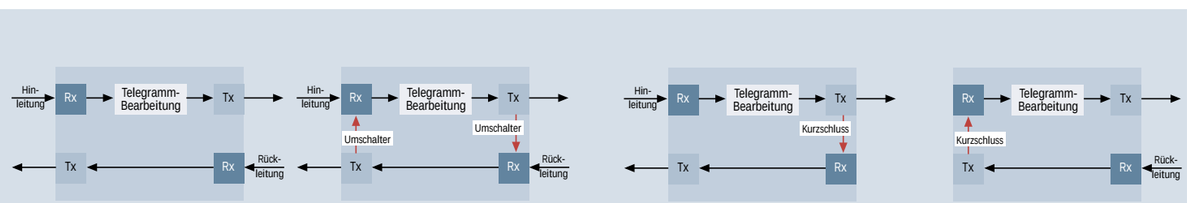
Da es sich bei den übertragenen Daten jederzeit um vollständig kompatible Ethernet-Telegramme handelt, kann die physikalische Übertragungsschicht an jeder Stelle und beliebig oft gewechselt werden. Bezogen auf das Beispiel mit den verschiedenen Schaltschränken und Maschinenmodulen, kann die jeweils kostengünstigste Übertragungsschicht genutzt werden. Innerhalb eines Schaltschrankes reicht der E-Bus; zwischen diesem und den Maschinenmodulen kommt die normale Ethernet-Kupfer-Physik mit bis zu 100 m zum Einsatz. Bei noch größeren Entfernungen oder extremen EMV-Belastungen kann wiederum an jeder Stelle auf LWL-Physik umgesetzt werden.

Einzige Voraussetzung an das Übertragungsmedium ist die Vollduplex-Fähigkeit, da EtherCAT so schnell reagiert, dass in der Regel bereits die Antwort zum Master zurückgesendet wird, während der Master noch die letzten Bytes seiner Anfrage absendet. Halbduplex-Übertragungsschichten, wie sie z. B. bei der Funkübertragung genutzt werden, würden dabei Kollisionen erzeugen und können für EtherCAT nicht eingesetzt werden. EtherCAT nutzt in der aktuellen Implementierung 100 MBaud Ethernet. Das Übertragungsprinzip ist davon aber unabhängig und kann zukünftig ohne Änderung auf 1 GBaud oder mehr angewendet werden.

Topologie

Die Topologie eines Kommunikationssystems ist mit ausschlaggebend für den erfolgreichen Einsatz in der Automatisierungsindustrie. Die Topologie hat großen Einfluss auf die Verkabelungsaufwendungen, Diagnose-Eigenschaften, Redundanz-Möglichkeiten und Hot-Plug-and-Play-Eigenschaften. Die bei Standard-Ethernet (100 Base TX) übliche Sternverkabelung hat zwar Vorteile bezüglich Hot-Plug-and-Play und Redundanz, die Verkabelungsaufwendungen und notwendigen Switche sind aber in verteilten Anwendungen mit vielen Teilnehmern kaum akzeptabel.

Bei EtherCAT stellen die Slaves logisch gesehen einen offenen Ringbus dar. Am offenen Ende schickt der Master, entweder direkt oder über Standard-Ethernet-Switche, Telegramme hinein und erhält sie am anderen Ende bearbeitet wieder zurück. Alle Telegramme werden vom ersten Teilnehmer an die nächsten weitergeleitet und vom letzten wird das Telegramm wieder zurück zum Master gesendet. Da ein normales Ethernet-Kabel bidirektional aufgebaut ist (getrennte Tx- und Rx-Leitungen) und auch alle EtherCAT-Slaves in der Rückrichtung übertragen können, ergibt sich ein physikalischer Strang.



Durch Abzweigungen, die prinzipiell an jeder Stelle möglich sind, kann aus der Strangstruktur eine flexible Baumstruktur aufgebaut werden. Eine Baumstruktur ermöglicht einfachste Verkabelungen; so können einzelne Äste z.B. in Schaltschränke oder Maschinenmodule verzweigen, während der Hauptstrang von einem Modul zum nächsten läuft.

Telegrammübertragung

Jeder EtherCAT-Slave hat zwei Rx- und Tx-Schnittstellen, über die die Telegramme jeweils in der Hinrichtung und in der Rückrichtung geleitet werden. Die Auswertung der Telegramme erfolgt immer in Hinrichtung, die Rückrichtung dient zum Verstärken und Regenerieren des Signals und zum Lokalisieren und Schließen von Leitungsunterbrechungen (Bild a).

Automatisches Öffnen und Schließen der Übertragungsleitung

Ein EtherCAT-Slave kann erkennen, ob auf seiner Hin- bzw. Rückleitung ein Trägersignal anliegt. Weiterhin besitzt jeder EtherCAT-Slave die Eigenschaft, den Datenfluss so umschalten zu können, dass ein über die Hinleitung empfangenes Telegramm sowohl über die Tx-Schnittstelle der Hinleitung als auch der Rückleitung gesendet werden kann. Und umgekehrt kann ein über die Rückleitung empfangenes Telegramm über die Tx-Schnittstelle der Rück- oder der Hinleitung geschickt werden (Bild b).

Wenn der EtherCAT-Slave auf seiner Rückleitung kein Trägersignal mehr erkennt, schaltet er die Rx-Schnittstelle der Hinleitung auf die Tx-Schnittstelle der Rückleitung kurz, d.h. ein über die Hinleitung empfangenes Telegramm wird bearbeitet und dann über die Tx-Schnittstelle der Rückleitung zurückgesendet. Durch diese Funktionalität werden keine zusätzlichen Abschlussmodule benötigt, da der EtherCAT-Slave automatisch erkennt, dass nach ihm kein weiterer Slave folgt.

Da dieser Zustand auch durch eine kurzzeitige Unterbrechung auftreten kann, wird auch im Kurzschlussfall versucht, ein Trägersignal bzw. das gerade im Durchlauf befindliche Telegramm über die Tx-Schnittstelle der Hinleitung zu senden, um eine wieder geschlossene Unterbrechung zu erkennen. Über die Rx-Schnittstelle der Hinleitung wird dann wieder ein Trägersignal erkannt und der Kurzschluss aufgehoben (Bild c).

Eine Unterbrechung kann natürlich auch auf der Hinleitung auftreten. Erkennt der EtherCAT-Slave an der Rx-Schnittstelle seiner Hinleitung kein Trägersignal mehr, so schließt er die Tx-Schnittstelle der Rückleitung mit der Rx-Schnittstelle der Hinleitung kurz. Über die Tx-Schnittstelle der Rückleitung wird in diesem Fall aber kein Trägersignal bzw. Telegramm mehr gesendet (Bild d).

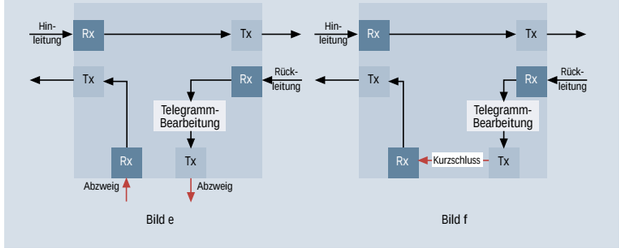
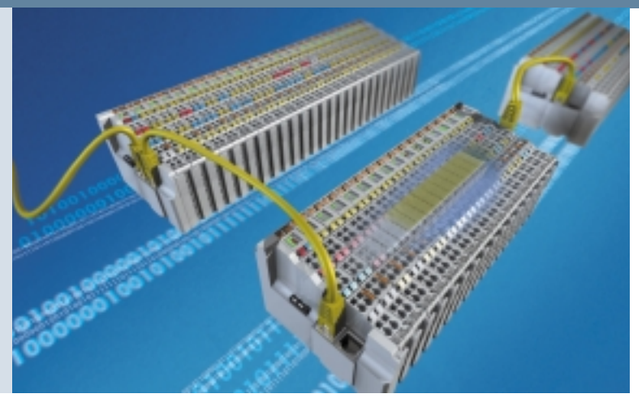


Bild e

Bild f



Abzweige in der Übertragungsleitung

Durch das automatische Erkennen eines Trägersignals können auch relativ einfach Abzweige realisiert werden, sofern diese auch über die Sende- und Empfangsfunktionalität eines EtherCAT-Slaves verfügen. In diesem Fall gibt es drei Rx- bzw. Tx-Schnittstellen (Hinleitung, Rückleitung und Abzweig). Die Hinleitung ist dabei immer kurzgeschlossen, d.h. ein Telegramm geht ohne Bearbeitung von der Rx-Schnittstelle der Hinleitung auf die Tx-Schnittstelle der Rückleitung. Die Rx-Schnittstelle der Rückleitung und die Tx-Schnittstelle des Abzweigs nehmen jetzt die Stellung der Hinleitung bei einem „normalen“ EtherCAT-Slave ein, die Rx-Schnittstelle des Abzweigs und die Tx-Schnittstelle der Rückleitung bekommen die Aufgaben der Rückleitung bei einem „normalen“ Slave (Bild e).

Solange kein Kabel an den Abzweig angeschlossen ist, werden Rx- und Tx-Schnittstelle der Rückleitung kurzgeschlossen, d.h. ein hierüber empfangenes Telegramm wird bearbeitet und über die Tx-Schnittstelle der RückLeitung gesendet (Bild f).

Sobald ein Kabel an den Abzweig angeschlossen ist, wird am Abzweig ein Trägersignal erkannt und der Kurzschluss aufgehoben. Da das Stecken oder Ziehen eines Abzweigs während des Betriebs erlaubt ist (im Gegensatz dazu stellt das Unterbrechen einer Linie einen Fehlerzustand dar), muss die Möglichkeit bestehen, das Stecken bzw. Ziehen rückwirkungsfrei durchführen zu können. Dazu gibt es ein Control-Bit im EtherCAT-Slave, mit dem das automatische Erzeugen bzw. Aufheben eines Kurzschlusses einer Übertragungsleitung deaktiviert werden kann. Außerdem verfügt jeder Abzweig über einen Schalter und eine LED, mit denen das Ziehen angefordert und bestätigt wird, so dass der Master die Möglichkeit hat, eine, der Durchlaufzeit durch den Abzweig entsprechende, zeitliche Lücke zu lassen und den Kurzschluss der Übertragungsleitung definiert durchzuführen.

Diagnose

Die Diagnosemöglichkeiten sind für den praktischen Einsatz eines verteilten Bussystems sicher ebenso entscheidend wie Performance-Daten, Topologie-Eigenschaften oder Verkabelungsaufwendungen. Auch hier erfüllt EtherCAT die Erwartungen an ein modernes Kommunikationssystem. Im Gegensatz zu Party-Line-Bussystemen (z.B. Profibus und CAN), bei denen die Teilnehmer alle am selben physikalischen Kabel hängen, und die Signale an alle Teilnehmer ohne Auffrischung gesendet werden, kommt bei Ethernet (zumindest ab 100 MBaud) und natürlich auch bei EtherCAT eine reine Punkt-zu-Punkt Übertragung zum Einsatz. Dadurch lassen sich Fehlerstellen oder auch nur sporadische Schwachstellen exakt lokalisieren, die sich bei Party-Line-Systemen entweder gar nicht oder nur mit speziellen Messaufbauten finden lassen.

Bruchstellendiagnose

Bruchstellen im logischen Kommunikationsring werden automatisch lokalisiert und geschlossen. Jeder Teilnehmer überwacht die Trägersignale sowohl auf der Hin- als auch auf der Rückrichtung und kann entsprechende Störungen feststellen. Wird ein Kabelbruch oder der Ausfall eines Nachbarn erkannt, schließt der Teilnehmer davor den Kommunikationsring, in dem er seine Hinrichtung mit der Rückrichtung kurzschließt. Dadurch entsteht immer ein kommunikationsfähiger geschlossener Ring, über den der Master die Bruchstelle exakt lokalisieren kann. Aufgrund der verteilten physikalischen Adressen bzw. der logischen Adressierung in den Teilnehmern ist auch weiterhin ein unveränderter Prozessdatenaustausch mit den noch verbleibenden Teilnehmern möglich.

Sicherungsmaßnahmen

Bei EtherCAT wird (mittels Checksumme) geprüft, ob ein Telegramm korrekt übertragen und von allen adressierten Teilnehmern (mittels Working-Counter) korrekt bearbeitet wurde. Als Checksumme wird die normale Ethernet-Checksumme benutzt, die sich am Ende des Ethernet-Telegramms befindet. Da ein oder mehrere Slaves das Telegramm während der Übertragung modifizieren, wird die Checksumme jedes Slaves neu berechnet. Bei einem erkannten Checksummenfehler wird ein Status-Bit im EtherCAT-Slave gesetzt und ggf. ein Interrupt zum Master ausgelöst, so dass eine Fehlerstelle genau lokalisiert werden kann.

Bei einer Leseoperation werden die adressierten Daten vom Slave in das dafür vorgesehene Datenfeld eingefügt und beim Schreiben entsprechend entnommen. In beiden Fällen inkrementiert der adressierte Slave einen Working-Counter, der sich am Ende jedes EtherCAT-Kommandos befindet. Da der Master weiß, wie viele Slaves von dem Telegramm adressiert sind, kann er anhand des Working-Counters erkennen, ob alle Slaves ihre Daten korrekt ausgetauscht haben.

Protokoll

An einen Feldbus werden unterschiedliche Anforderungen gestellt, was die Übertragungseigenschaften von Daten betrifft. Parameterdaten werden azyklisch und in großen Mengen übertragen, wobei die zeitlichen Anforderungen relativ unkritisch sind, und die Übertragung in der Regel von der Steuerung angestoßen wird. Diagnosedaten werden ebenfalls azyklisch und ereignisgesteuert übertragen; die zeitlichen Anforderungen sind aber höher und die Übertragung wird in der Regel von einem Peripheriegerät angestoßen. Prozessdaten dagegen werden zyklisch mit verschiedenen Zykluszeiten übertragen. Dabei ist es wichtig, dass es nicht zu Zyklusausfällen kommt. Die zeitlichen Anforderungen sind bei der Prozessdatenkommunikation am höchsten.

Für die unterschiedlichen Kommunikationsarten gibt es bei EtherCAT entsprechende Adressierungsmöglichkeiten, die den Anforderungen jeweils optimal entsprechen.

Adressierung

Physikalische Adressierung: Mit der physikalischen Adressierung wird ein Teil des 64 kByte Adressraums eines Slaves gelesen oder geschrieben. Mit einem Telegramm ist immer genau ein Slave adressiert. Diese Adressierung wird besonders zur Übertragung von Parameterdaten genutzt. Zur Adressierung des Slaves gibt es zwei Varianten, die Auto-Increment- sowie die Fixed-Address-Adressierung die im Folgenden beschrieben werden.

Auto-Increment: Mit der Auto-Increment-Adressierung kann jeder Slave anhand seiner Position im Kommunikationsring adressiert werden. Dabei inkrementiert jeder Slave das 16-Bit-Adressfeld während des Telegrammdurchlaufs; adressiert ist der Slave, der ein Adressfeld mit dem Wert 0 empfängt. Will die Steuerung den zehnten Teilnehmer im Ring ansprechen, sendet sie ein Telegramm mit Auto-Increment-Adressierung mit dem Startwert 9, der pro Teilnehmer um eins inkrementiert wird. Die Auto-Increment-Adressierung wird in der Regel nur in der Startup-Phase benutzt, in der der Master Stationsadressen an die Slaves verteilt. Danach können sie unabhängig von ihrer Position im Ring adressiert werden.

Diese Vorgehensweise bietet den Vorteil, dass bei den Slaves keine Stationsadressen manuell einzustellen sind. Ein nachträgliches Einfügen von Slaves führt nicht zu neuen Adressen.

Fixed-Address: Bei der Fixed-Address-Adressierung werden die Slaves über die vom Master in der Startup-Phase verteilte Stationsadresse angesprochen. Damit wird gewährleistet, dass auch bei einer Veränderung des Rings während des Betriebs die Slaves über die gleiche Adresse angesprochen werden können.

Logische Adressierung: Bei der logischen Adressierung wird kein Slave einzeln sondern ein Teil eines 4 GByte großen logischen Adressraums angesprochen, der sich über beliebig viele Slaves verteilen kann. Die Zuordnung von physikalischen Adressen eines Slaves zu logischen Adressen im EtherCAT-Bus wird im Master projektiert und in der Startup-Phase an die Fieldbus-Memory-Management-Units (FMMU) der Slaves übertragen. Eine FMMU hat die Aufgabe, den physikalischen Adressen eines Teilnehmers eine logische Adresse zuzuordnen. Je FMMU wird eine logische, bitorientierte Startadresse, eine physikalische Startadresse, eine Bit-Länge sowie ein Typ, der angibt in welcher Richtung gemappt werden soll (Inputs oder Outputs), konfiguriert. Damit ist es möglich, beliebige Daten eines EtherCAT-Slaves bitweise auf beliebige logische Adressen zu mappen. Der Slave überprüft beim Empfang eines Telegramms mit logischer Adressierung, ob eine seiner FMMUs einen Adressmatch hat, und fügt ggf. Daten an der entsprechenden Stelle des Datenfeldes in das Telegramm ein (Typ Inputs) bzw. entnimmt Daten von der entsprechenden Stelle aus dem Telegramm (Typ Outputs). Dadurch ist es möglich, Telegramme flexibel zusammenzustellen und an die Bedürfnisse der Steuerung optimal anzupassen. Damit eignet sich diese Adressierungsart besonders für die Übertragung zyklischer Prozessdaten.

Multiple-Adressierung: Mit der Multiple-Adressierung können physikalische Adressbereiche mehrerer Slaves gelesen werden. Der erste Slave wird dabei per Stationsadresse adressiert und schaltet das Multiple-Read-Flag im Telegramm ein, über das die im Ring folgenden Slaves erkennen können, dass sie adressiert sind. Jeder Slave, der über den geforderten physikalischen Adressbereich verfügt, trägt seine Stationsadresse und die entsprechenden Daten in das Datenfeld ein, solange noch Platz ist.

Dazu wird in den ersten beiden Bytes des Datenfeldes ein Working-Pointer übertragen, der von jedem Slave, der Daten in das Datenfeld eingefügt hat, inkrementiert wird. Damit ist es möglich, spezielle Diagnosedaten im Slave zu definieren, die mit der Multiple-Adressierung in der Regel mit einem Telegramm abgeholt werden können.

Diese Adressierungsart eignet sich besonders für die Übertragung von azyklischen Diagnosedaten, da flexibel und schnell ausgewählte Daten von mehreren Teilnehmern gelesen werden können.

Interrupts: In jedem Telegramm befindet sich ein 16-Bit-Interruptfeld, mit dem sehr schnell Ereignisse gemeldet werden können. Mit diesem Interruptfeld können bis zu 16 Diagnoseadressen in den Slaves verknüpft werden. Wenn bei einem oder mehreren Slaves das entsprechende Ereignis auftritt, wird im nächsten empfangenen Telegramm von jedem Slave, bei dem das Ereignis aufgetreten ist, das entsprechende Interrupt-Bit gesetzt. Der Master kann daraufhin durch Senden eines Telegramms mit Multiple-Adressierung die zugehörigen Diagnosedaten der entsprechenden Slaves lesen.

Mit den Interrupts können Ereignisse ohne Zeitverzögerung gemeldet werden. In Kombination mit der Multiple-Adressierung werden Diagnosedaten sehr schnell ereignisgesteuert übertragen, ohne die Übertragung zeitkritischer Prozessdaten zu stören, da Sendezeitpunkt und Größe der zusätzlichen Telegramme vom Master gesteuert werden können.

Broadcast: Mit einem Broadcast-Write können physikalische Adressbereiche aller Slaves beschrieben werden; so lassen sich z. B. Zustandsübergänge bei allen Slaves gleichzeitig durchführen. Mit einem Broadcast-Read werden physikalische Adressbereiche aller Slaves verodert gelesen. Wenn eine Anwendung nur sinnvoll arbeiten kann, wenn sich alle Slaves im Okay-Zustand befinden, kann z. B. dieser Okay-Zustand sehr einfach und schnell mit einem Broadcast-Read von allen Teilnehmern abgefragt werden.

Telegrammaufbau

Es wird ein Standard-Ethernet-Telegramm mit einem speziellen Ether-Type benutzt. Innerhalb des Datenfeldes des Ethernet-Telegramms befinden sich ein oder mehrere EtherCAT-Kommandos. Jedes EtherCAT-Kommando hat einen Header von 10 + 2 Bytes, der wie folgt aufgebaut ist:

Byte	Bezeichn.	Bedeutung
1	**CMD**	Kommando, identifiziert die im Folgenden beschriebenen Kommandos
2	**IDX**	Index, wird vom Slave unverändert weitergesendet,
		damit kann der Master das Telegramm beim Empfang wieder leicht zuordnen
3,4	**ADP**	Address-Page, abhängig vom verwendeten Kommando
5,6	**ADO**	Address-Offset, abhängig vom verwendeten Kommando
7,8	**LEN**	Bit 0-10: Länge des Datenfeldes DATA, Bit 11-15: Flags
9,10	**INT**	Interrupt-Feld
11-(n-2)	**DATA**	Datenfeld
(n-1),n	**CNT**	Working-Counter, wird von jedem Slave inkrementiert, der adressiert war und
		das Telegramm erfolgreich bearbeiten konnte

Kommando	ADP	ADO	Beschreibung
NOP	-	-	Keine Bearbeitung
APRD	Das ADP-Feld wird	Das ADO-Feld enthält die	Lesen eines phys. Bereiches
APWR	von jedem Slave inkrementiert;	physikalische Adresse innerhalb	mit Auto-Increment-Adressierung
APRW	adressiert ist der Slave,	des Slaves, ab der sich die über-	Schreiben eines phys. Bereiches
	der beim Empfang	tragenen Daten befinden.	mit Auto-Increment-Adressierung
	den Wert 0 feststellt.		Lesen und Schreiben eines
			phys. Bereiches mit Auto-Increment-Adressierung
FPRD	Das ADP-Feld enthält die		Lesen eines phys. Bereiches mit Fixed-Adressierung
FPWR	Stationsadresse des		Schreiben eines phys. Bereiches mit Fixed-Adressierung
FPRW	adressierten Slaves.		Lesen und Schreiben eines
			phys. Bereiches mit Fixed-Adressierung
BRD	Jeder Slave ist adressiert		Verodertes Lesen eines phys. Bereiches bei allen Slaves
BWR	und inkrementiert das ADP-Feld.		Schreiben eines phys. Bereiches bei allen Slaves
BRW			Verodertes Lesen und Schreiben eines phys. Bereiches
LRD	Das ADP-Feld enthält das	Das ADO-Feld enthält das	Lesen eines logischen Speicherbereiches
LWR	Hi-Word der logischen 32-Bit-	Lo-Word der logischen 32-Bit-	Schreiben eines logischen Speicherbereiches
LRW	Adresse, ab der sich die	Adresse, ab der sich die	Lesen und Schreiben eines logischen Speicherbereiches
LRO	übertragenen Daten befinden.	übertragenen Daten befinden.	Verodertes Lesen eines logischen Speicherbereiches
MRD	Das ADP-Feld enthält die	Das ADO-Feld enthält	Lesen eines Speicherbereiches bei mehreren Slaves.
	Stationsadresse des ersten	die physikalische Adresse	In den ersten beiden Bytes des Datenfeldes
	adressierten Slaves, der	innerhalb des Slaves, ab der sich	befindet sich der Working-Pointer, der auf das
	das MRD-Flag setzt. Ein Slave ist	die übertragenen Daten befinden.	nächste freie Datum innerhalb des Datenfeldes
	adressiert, wenn die Stationsadresse		zeigt. Nachdem ein Slave Daten in das Datenfeld
	übereinstimmt oder das MRD-Flag		eingefügt hat, inkrementiert er den Working-Pointer
	gesetzt ist und der mit dem ADO-		um die entsprechende Länge. Ein Slave fügt nur
	Feld und den MRD-Längen-Flags		Daten in das Telegramm ein, wenn er adressiert
	adressierte phys. Bereich definiert ist.		ist und die einzufügenden Daten noch in
			das Telegramm passen (Working-Pointer +
			MRD-Längen-Flags < Länge des Datenfeldes).

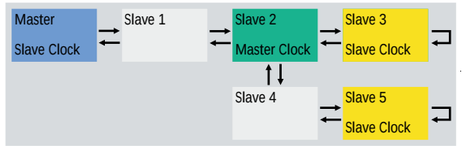
EtherCAT-Kommandos

In der Tabelle sind die von EtherCAT-Teilnehmern unterstützten Kommandos aufgeführt, die direkt in Hardware im EtherCAT-Asic implementiert sind. Letztendlich ergeben sich reine Lese- und Schreibbefehle in den 64-kByte-Adressraum der Teilnehmer. Die Kommandos sind mit den beschriebenen Adressierungsarten kombiniert und liefern praxisgerechte, bandbreitenschonende Zugriffsmechanismen für alle Anforderungen an das Kommunikationssystem.

Distributed-Clocks

Mit den Distributed-Clocks ist es bei EtherCAT möglich, in allen Busteilnehmern die gleiche Uhrzeit zu haben. Es gibt einen ausgewählten Teilnehmer, der die Master-Clock besitzt, auf die sich die Slave-Clocks der anderen Teilnehmer und der Steuerung synchronisieren. Dazu sendet die Steuerung in gewissen Abständen (so häufig, dass die Slave-Clocks innerhalb der Grenzen nicht auseinander laufen) ein spezielles Telegramm, in das der Busteilnehmer mit der Master-Clock seine aktuelle Uhrzeit einträgt. Das Telegramm wird dann von den Busteilnehmern mit Slave-Clock aus demselben Telegramm gelesen. Dies ist auf Grund der Ringstruktur von EtherCAT möglich, wenn die Master-Clock vor den Slave-Clocks angeordnet ist.

Das EtherCAT-Asic bietet dem angeschlossenen Microcontroller darüber hinaus noch die Funktionalität, mittels einer Capture/Compare-Einheit seinen lokalen Timer mit der Slave-Clock abzugleichen.



In der Abbildung sind die Distributed-Clocks bei Slave 1 und 4 nicht aktiviert, Slave 2 ist vor Slave 3 und Slave 5 im Ring positioniert und ist daher die Master-Clock.

Da pro Slave auf dem Hin- und Rückweg eine geringe Verzögerung – sowohl im Teilnehmer als auch in der dazwischen liegenden Übertragungsstrecke – besteht, ist die Laufzeit zwischen Master-Clock und jeweiliger Slave-Clock bei der Synchronisierung der Slave-Clocks zu berücksichtigen. Zur Messung der Laufzeit sendet der Master einen Broadcast-Read auf eine spezielle Adresse, womit jeder Slave veranlasst wird, den Empfangzeitpunkt des Telegramms (bez. seiner lokalen Clock) auf dem Hin- und auf dem Rückweg zu speichern. Diese gespeicherten Zeitpunkte können vom Master eingelesen und entsprechend verrechnet werden.

Querverkehr

Auch wenn EtherCAT ein klares Master-Slave-Kommunikationsmodell nutzt und damit ideal die hierarchische Steuerungstechnik unterstützt, lässt sich mit den Eigenschaften der Feldbus-Memory-Management-Unit (FMMU) sehr einfach der Querverkehr zwischen EtherCAT-Teilnehmern realisieren. Hierzu werden Speicherbereiche aus dem 4 GByte großen logischen Adressraum für Querverkehr reserviert und vom Master zyklisch ausgetauscht. Der Master stellt alternierend eine Leseanfrage und im nächsten Zyklus einen Schreibauftrag für den entsprechenden Speicherbereich. Alle entsprechend konfigurierten Teilnehmer blenden ihre Querverkehrsdaten ein bzw. entnehmen sie im nächsten Zyklus. Für den Master sind diese Daten transparent, und er sorgt nur für den zyklischen Austausch.

Im Vergleich zu Partyline-Bussystemen, an denen alle Teilnehmer am selben Kommunikationsmedium hängen, vergeht zwar ein Zyklus; dies wird aber durch die überragende Nutzdatenrate und die dadurch möglichen kurzen Zykluszeiten mehr als wettgemacht. Die beschriebene Vorgehensweise hat auch den Vorteil, dass die Querverkehrsdaten von mehreren Quellen eingesammelt werden und dann im nächsten Zyklus bei allen angesprochenen Senken gleichzeitig ankommen. Bei einer Zykluszeit von z. B. 100 µs können bereits ca. 1000 Bytes von fast beliebig vielen Quellen an ebenso viele Senken gesendet werden.

Noch wichtiger ist, dass die vom Master geplante und von ihm initiierte Querkommunikation absolut deterministisch erfolgt und daher der zyklische Prozessdatenaustausch nicht beeinflusst und unnötig Bandbreite reserviert werden muss.

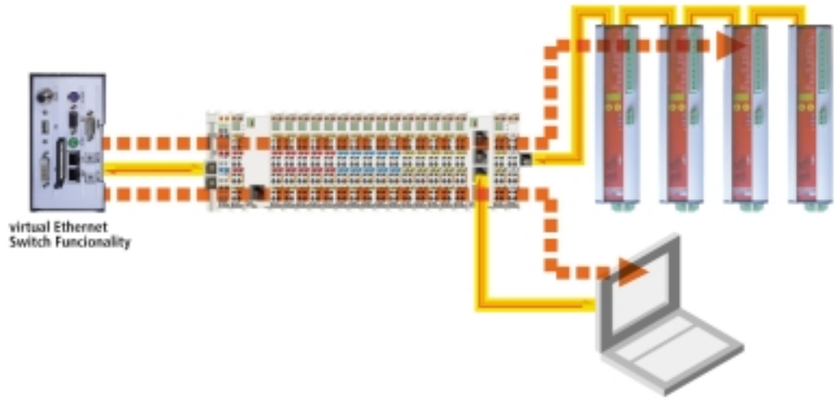
Ethernet over EtherCAT

Neben den bereits beschriebenen EtherCAT-Adressierungsarten, über die mit den EtherCAT-Teilnehmern kommuniziert wird, werden auch Standard-IP-basierte Protokolle wie TCP/IP, UDP/IP und alle darauf basierenden höheren Protokolle (HTTP, FTP, SNMP etc.) von einem Ethernet-Feldbus erwartet. Sinnvollerweise sollten dabei einzelne Ethernet-Telegramme transparent übertragen werden, da dadurch keine Einschränkungen bezüglich der zu übertragenden Protokolle entstehen.

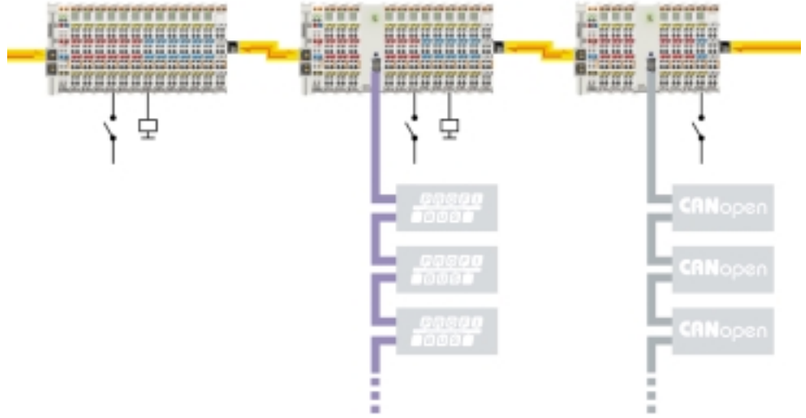
Prinzipiell gibt es zwei unterschiedliche Ansätze, um azyklische Ethernet-Telegramme im zyklischen Feldbusbetrieb zu übertragen. Die erste Variante besteht darin, im Zyklus eine entsprechende Zeitscheibe zu reservieren, in der die azyklischen Ethernet-Telegramme eingebettet werden können. Diese Zeitscheibe muss entsprechend groß gewählt werden, dass vollständige Ethernet-Telegramme darin Platz finden. Die übliche MTU (Maximum Transmission Unit) liegt bei 1514 Byte, was in etwa 125 µs bei 100 MBaud entspricht. Daraus ergibt sich für Systeme, die diese Variante nutzen, eine minimale Zykluszeit von etwa 200-250 µs. Eine Verringerung der MTU führt häufig zu Problemen: Das IP-Protokoll lässt die Fragmentierung zwar prinzipiell zu; es wird aber davon abgeraten, und es wird sie in der nächsten Generation (IPv6) nicht mehr geben. Insbesondere bei UDP/IP-Übertragungen können zudem Datenkonsistenzprobleme auftreten.

EtherCAT nutzt die zweite mögliche Variante, bei der Ethernet-Telegramme getunnelt und im entsprechenden Teilnehmer wieder zusammengestellt werden, bevor sie als vollständige Ethernet-Telegramme weitergeleitet werden. Dieses Verfahren schränkt die erreichbare Zykluszeit nicht ein, da je nach verfügbarer Bandbreite die Fragmente (EtherCAT statt IP-Fragmentierung) optimal eingestellt werden können. EtherCAT definiert dabei einen Standardkanal, der es prinzipiell jedem EtherCAT-Teilnehmer ermöglicht, am normalen Ethernet-Verkehr teilzuhaben. So kann zum Beispiel in einem intelligenten Antriebsregler, der mit 100 µs Zykluszeit seine Prozessdaten austauscht, ein HTTP-Server integriert werden, der seine eigene Diagnoseschnittstelle in Form von Web-Seiten mitbringt.

Eine weitere Anwendung der übertragenen Ethernet-Telegramme realisieren die Hub- und Switchklemmen. Diese bieten an beliebiger Stelle im EtherCAT-System normale Ethernet-Ports mit entsprechenden RJ45-Buchsen, über die jedes Ethernet-Gerät angeschlossen werden kann. Dies kann z. B. ein Service-Rechner sein, der direkt mit der Steuerung kommuniziert, die Web-Seite eines intelligenten EtherCAT-Teilnehmers abfragt oder einfach über die Steuerung ins Intra- oder Internet routet. Die Switchklemme integriert zusätzlich zur Hubklemme einen



Ethernet over EtherCAT: Hub- und Switchklemmen



Spezielle EtherCAT-Teilnehmer: Feldbusmaster

An der EtherCAT-Entwicklung waren folgende Beckhoff-Mitarbeiter beteiligt:
Franz-Joseph Kucharski, Thorsten Bunte, Thomas Rettig, Manfred Ullenbrock, Michael Schlegel, Holger Büttner, Dirk Janssen sowie der Geschäftsführer Hans Beckhoff

Switchbaustein und bietet dadurch mehrere Ethernet-Anschlüsse in einem Gerät an. Im EtherCAT-Master ist ebenfalls eine Switchfunktionalität in Software integriert, die für das Routing der einzelnen Ethernet-Telegramme von und zu den EtherCAT-Teilnehmern und dem IP-Stack des Hostbetriebssystems zuständig ist. Die Switchfunktionalität ist dabei identisch mit der eines üblichen Layer-2-Switches und reagiert protokollunabhängig auf die verwendeten Ethernet-Adressen.

Feldbusanschaltungen

Dank der Leistungsfähigkeit von EtherCAT können auch komplexe Busteilnehmer, wie z. B. ein Feldbusmastermodul, realisiert werden. In der Steuerung werden keine zusätzlichen Anschaltungen mehr benötigt. EtherCAT arbeitet quasi als PCI-Ersatz, in der Regel sogar mit Performance-Gewinn.

Feldbusmasteranschaltungen in Steuerungen haben eine Prozessabbildschnittstelle zum Austausch von zyklischen Prozessdaten und eine Mailbox-Schnitttstelle zur Übergabe von azyklischen Parameter- oder Diagnosedaten. Das Prozessabbild kann über EtherCAT in der Regel mit einem einzigen Telegramm übertragen werden, wobei der Master das Output-Prozessabbild sendet und das Inputprozessabbild wieder empfängt.

Die Grenze liegt bei jeweils 1488 Bytes Input- und Output-Prozessdaten, die neben Ethernet- und EtherCAT-Header noch im Ethernet-Frame verfügbar sind. Die Übertragung dauert maximal ca. 125 µs, der zusätzliche Aufwand in der Steuerung ist verhältnismäßig gering, da der Ethernet-Chip selbständig per DMA auf den Speicher zugreift, ohne den Steuerungsprozessor zu belasten. Herkömmliche Feldbusanschaltungen haben in der Regel keine DMA-Schnittstelle zum Speicher: Die Übertragung von je 1488 Bytes Input- und Output-Prozessabbild dauert bei einer Feldbusanschaltung mit 16-Bit-Interface über PCI ca. 500 µs. Die Übertragungsdauer bei dezentralen EtherCAT-Modulen im Vergleich zu herkömmlichen 16-Bit-PCI-Anschaltungen reduziert sich also um bis zu 70 %.

Auch eine äquidistante Schnittstelle, wie sie z. B. bei Antriebsregelungen über Sercos, Profibus DP-V2 oder CANopen mit TwinCAT üblich ist, lässt sich problemlos realisieren. Über die Distributed-Clocks kann der Zyklus des Feldbusmasters nahezu jitterfrei auf den EtherCAT-Zyklus synchronisiert werden. Auch ohne Distributed-Clocks sorgt eine PLL in dem Feldbusmastermodul dafür, dass ein Jitter des EtherCAT-Telegramms ausgeregelt wird.

Auf die Mailboxschnittstelle eines Feldbusmastermoduls kann einfach über physikalische Adressierung zugegriffen werden. Diagnosedaten können über die Interrupt-Bits gemeldet und per Multiple-Read-Adressierung übertragen werden.

Asic-Schnittstellen

Um sowohl ganz einfache Teilnehmer, wie z. B. ein 2-Bit digitales Eingangsmodul als auch komplexe Teilnehmer, wie ein Feldbusmastermodul gleichermaßen kosten- und performance-optimal zu implementieren, sind verschiedene Asic-Schnittstellen notwendig.

4-Bit-In/Out: Das 4-Bit-In/Out-Interface ist für digitale Ein- und Ausgänge mit bis zu 4 Bit Daten gedacht und stellt eine besonders kostengünstige Anschaltung dar. Das Asic hat 4 konfigurierbare Input/Output-Pins, es ist kein Microcontroller notwendig.

SSI-Slave: Das SSI-Slave-Interface ist für alle Geräte mit wenigen Prozessdatenbytes geeignet, wie z.B. analoge Module, Geber, Encoder oder auch einfache Antriebe. Der Asic verfügt über einen 4 kByte großen Speicher, der über EtherCAT adressiert werden kann. Die Datenübertragung vom Asic zum Microcontroller erfolgt über ein bis zu 10 MBit schnelles SSI-Interface, das vom Buszugriff entkoppelt ist. Die SSI-Übertragung stellt eine kostengünstige Variante für Geräte mit wenigen Prozessdatenbytes dar. Die Menge der azyklischen Parameter- und Diagnosedaten ist nahezu unbeschränkt, sofern deren Übertragung nicht zeitkritisch ist.

32-Bit-Parallel-In/Out: Das 32-Bit-Parallel-In/Out-Interface benötigt ein größeres Pinout als die beiden bereits beschriebenen Schnittstellenvarianten und eignet sich für den Anschluss von bis zu 32 Bit digitalen Inputs bzw. Outputs, aber auch für einfache Aktoren oder Sensoren, die mit 32 Datenbits auskommen. Im Gegensatz zum SSI-Slave-Interface ist der Zugriff auf die Prozessdaten wesentlich schneller, und es ist kein Microcontroller notwendig, allerdings ist das Datenaufkommen auch auf die 32 Bits beschränkt.

Paralleles 16-Bit-Microcontroller-Interface: Das parallele 16-Bit-Microcontroller-Interface entspricht herkömmlichen Schnittstellen bei Feldbus-Asics mit DPRAM-Schnittstelle. Diese Anschaltung ist etwas aufwändiger aber dafür auch leistungsfähiger als die anderen Asic-Schnittstellen und eignet sich daher insbesondere für komplexe Teilnehmer mit größerem Datenaufkommen.

Fazit

In der Automatisierungstechnik zeichnet sich der Trend ab, Ethernet auch in der Feldebene einzusetzen. Unterschiedliche Ansätze versprechen hohe Bandbreiten, geringe Kosten, vereinfachte vertikale Integration, Einsatz von Standardkomponenten aus der Bürowelt und geringe Konfigurations- und Diagnoseaufwendungen. Das Ganze kombiniert mit der nötigen Echtzeitfähigkeit.

Bei genauerem Hinsehen weichen viele der Argumente auf oder verkehren sich ins Gegenteil: Die vergleichsweise hohe Bandbreite von 100 MBaud Ethernet wird bei typischen I/O-Knoten mit wenigen Bytes Prozessdaten zunichte gemacht. Ein Teilnehmer, der z. B. vier Bytes pro Richtung hat, kommt auf eine Nutzdatenrate von 3-4 Prozent. Auch die Kostenseite spricht eher gegen den Einsatz im Feld. Neben den reinen Anschaltungskosten kommt noch ein recht hoher Bedarf an Rechenleistung für die Verarbeitung der Telegramme hinzu. Der Einsatz von Standardkomponenten hört meisten dann auf, wenn ein gewisses Maß an Echtzeitfähigkeit gefordert wird. Außerdem ist die typische geswitchte Sternverkabelung eher ungünstig für den Einsatz im Feld. Selbst die Konfiguration wird nicht leichter: Die Vergabe der notwendigen IP-Adressen wird nicht selten in Konflikten mit der IT-Abteilung enden.

EtherCAT geht hier andere Wege und verbindet die Vorteile aus der Feldbustechnik mit den ansonsten unbestreitbaren Vorteilen der Ethernet-Welt. Die zur Verfügung stehende Bandbreite wird nahezu vollständig ausgenutzt, die Kosten reduzieren sich auf eine einfache Asic-Anschaltung im EtherCAT-Teilnehmer. Standardkomponenten werden dort eingesetzt, wo sie auch wirklich Standard sind – in der Steuerung und nicht in der 2-Bit-I/O-Klemme. EtherCAT benötigt keine IP-Adressen, die Konfiguration läuft nach einfachen Algorithmen vom Master gesteuert automatisch ab. Die vertikale Integration ist aber trotzdem gegeben. Teilnehmer, die eine IP-Adresse haben möchten, bekommen diese auch und sind dann vollständig transparent im Netzwerk integriert.

Mit EtherCAT lassen sich hochperformante Maschinensteuerungen realisieren, bei denen viele verteilte Signale mit Zykluszeiten deutlich unter 100 µs ausgetauscht werden können. Das System eignet sich aber ebenso gut für kostengünstige Steuerungsapplikationen, die mit drei Größenordnungen größeren Zykluszeiten auskommen, wie z.B. mit 100 ms in der Gebäudeautomatisierung. Jeder handelsübliche PC oder jeder Steuerungscontroller mit integriertem Ethernet-Port kann dabei als Master eingesetzt werden. EtherCAT bietet daher eine einheitliche, leistungsfähige Kommunikationsbasis für die gesamte Automatisierungstechnik. Von der „Klein“-SPS für weniger als 100 € bis zur Hochleistungs-CNC kann die gleiche Systemtechnik eingesetzt werden.