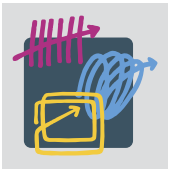
Bild: Ethercat Technology Group



Halle 6
Stand 208

Bild1: Die Verfügbarkeit industrieller Netzwerke ist zu einem kritischen Faktor bei der Verfügbarkeit der Maschinen und Anlagen geworden. Aus diesem Grund sind leistungsfähige Diagnosefunktionen von größter Bedeutung.

Diagnose mit Ethercat – Teil 1

Präzise Fehlerdiagnose

Neben zentralen Features wie etwa hoher Performance und Schnelligkeit spielt bei der Wahl einer Kommunikationstechnologie für das industrielle Umfeld auch die Verfügbarkeit von Werkzeugen zur Netzwerkdiagnose eine tragende Rolle. Ethercat verfügt systeminhärent über eine Vielzahl an Diagnoseeigenschaften, mithilfe derer Fehler im System exakt erkannt und lokalisiert werden können.

Die Diagnosefunktionen eines Feldbusses sollten zwei grundsätzliche Bedingungen erfüllen:

- Schnelle Reaktion: Die Steuerung sollte Anomalitäten im Netzwerk augenblicklich erkennen, um schnellstmöglich reagieren und so mechanischen Schaden, Material- oder Synchronitätsverlust innerhalb der Applikation verhindern zu können.
- Präzise Analyse: Das Netzwerk sollte der Steuerung detaillierte Statusinformationen zur Verfügung stellen, damit der Master bestimmen kann, ob Fehler durch EMV-Störungen, Hardware-Ausfälle oder Firmware-Verhalten verursacht wurden.

Dank seiner Protokolleigenschaften sowie der Slave-Struktur erfüllt Ethercat beide dieser Anforderungen gleichermaßen.

Der Ethercat-Frame: Struktur und Weiterleitung

Ethercat nutzt Standard-Ethernet gemäß IEEE802.3, wobei die Nutzlast der Ethercat-Frames aus einer variablen Anzahl sogenannter Ethercat-Datagramme besteht (Bild 2). Jeder Ethercat-Frame durchläuft das Netzwerk in eindeutiger Reihenfolge, verteilt Ausgangs- und sammelt Eingangsdaten in den Slaves und kehrt dann zum Master zurück (Bild 3). Auf Slave-Seite wird die Verarbeitung und Weiterleitung von einer speziellen Komponente, dem sogenannten Ethercat Slave Controller (ESC), durchgeführt, wie in Bild 4 dargestellt. Ports lenken die Frames in die korrekte Richtung, während die Ethercat-Zentraleinheit Daten von bzw. zu den Datagrammen sowie dem Slave-Speicher liest bzw. schreibt. Der ESC-Speicher als Teil der 'Ethercat Processing Unit' enthält sowohl Applikationsdaten (Prozessdaten, Mailbox) als auch Register: Verschiedene Standardregister sind in diesem Bereich sowohl für die Hardware- als auch für die Firmware-Diagnose definiert. Jedes Datagramm spezifiziert, welcher Vorgang im Slave-Speicher durch einen Kommando-Code ausgeführt werden soll. Bild 5 zeigt den Datagramm-Aufbau.

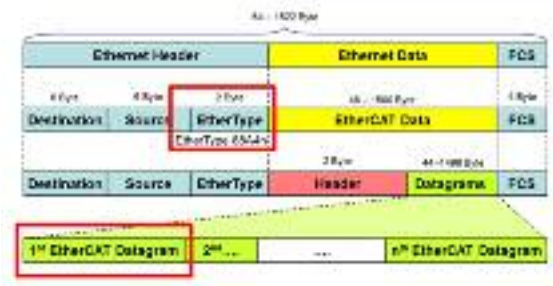


Bild 2: Struktur eines Ethercat-Frames

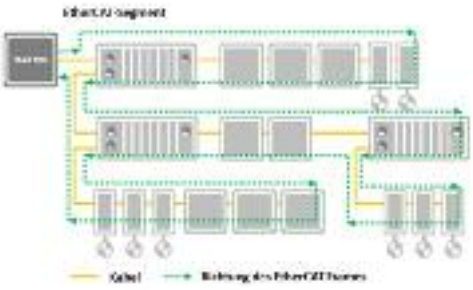
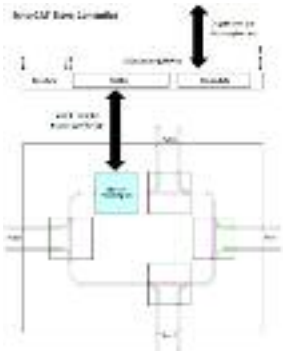


Bild 3: Verarbeitungsreihenfolge der Ethercat-Frames



Bilder: Ethercat Technology Group

Bild 4: Interne Struktur des Ethercat-Slave-Controllers

Ethercat definiert 15 verschiedene Kommando-Codes, die in der Tabelle unten aufgeführt sind. Jedes Ethercat-Datagramm, und damit jedes Ethercat-Kommando, hat am Ende einen sogenannten Working Counter. Mit diesem kann die Masterapplikation noch im selben Zyklus prüfen, ob der Datenaustausch gelungen ist. Daher gilt der Working Counter als äußerst effektiver Überwachungsmechanismus.

Working Counter

Der Working Counter besteht aus einem 16Bit-Feld und befindet sich am Ende eines jeden Ethercat-Datagramms (Bild 6). Wird ein Frame vom Master versendet, so steht der Working Counter für sämtliche Ethercat-Datagramme auf null. Wenn ein Slave von einem speziellen Datagramm adressiert ist und das entsprechende Kommando verarbeitet wurde (also wenn der Slave-Speicher gelesen bzw. beschrieben werden konnte), wird der zugehörige Working Counter in vordefinierter Weise vom ESC inkrementiert:

Aufgrund der Netzwerkkonfiguration ‘kennt’ der Master den erwarteten Wert des Working Counters für jedes Datagramm und kann diesen mit dem tatsächlichen Wert des zurückkehrenden Frames vergleichen. Working

Code	Name	Beschreibung
NOP	No Operation	Slave ignoriert den Befehl
APRD, APWR, APRW	Auto Increment Read, Auto Increment Write, Auto Increment Read Write	Jeder Slave inkrementiert die Adresse um 1. Der Slave, der das Datagramm mit der Adresse 0 empfängt, verarbeitet die Daten.
FPRD, FPWR, FPRW	Configured Address Read, Configured Address Write, Configured Address Read Write	Der Slave, dessen konfigurierte Adresse zu der Datagrammadresse passt, verarbeitet die Daten.
BRD, BWR, BRW	Broadcast Read, Broadcast Write, Broadcast Read Write	Alle Slaves inkrementieren die Adresse um 1. Alle Slaves verarbeiten die Daten.
LRD, LWR, LRW	Logical Memory Read, Logical Memory Write, Logical Memory Read Write	Der Slave, dessen logische Adresse (konfiguriert im entsprechenden FMMU-Objekt) zur Datagrammadresse passt, verarbeitet die Daten.
ARMW	Auto Increment Read Multiple Write	Jeder Slave inkrementiert die Adresse um 1. Der Slave, der das Kommando mit der Adresse 0 empfängt, fügt dem Datagramm Eingangsdaten hinzu, alle anderen entnehmen dem Datagramm an derselben Stelle Ausgangsdaten.
FRMW	Configured Address Read Multiple Write	Der Slave, dessen konfigurierte Adresse zur Datagrammadresse passt, fügt dem Datagramm Eingangsdaten hinzu, alle anderen entnehmen dem Datagramm an derselben Stelle Ausgangsdaten.

Tabelle 1: Ethercat definiert 15 verschiedene Kommando-Codes.

Kommandotyp	Ergebnis	Inkrementierung
Lesen	nicht erfolgreich gelesen (oder Slave nicht adressiert)	nein
	erfolgreich gelesen	+1
Schreiben	nicht erfolgreich geschrieben (oder Slave nicht adressiert)	nein
	erfolgreich geschrieben	+1
Lesen + Schreiben	nicht erfolgreich gelesen oder geschrieben (oder Slave nicht adressiert)	nein
	erfolgreich gelesen	+1
	erfolgreich geschrieben	+2
	erfolgreich gelesen und geschrieben	+3

Tabelle 2: Der ESC inkrementiert den Working Counter in vordefinierter Weise.

Counter-Fehler können durch verschiedene Ereignisse verursacht werden, die verhindern, dass ein oder mehrere Slaves ein Datagramm korrekt verarbeiten: Etwa wenn ein Slave ausgeschaltet oder vom Netzwerk getrennt wird, auf Grund eines internen Firmware-Fehlers in einen anderen Status wechselt, in welchem gewisse Kommandotypen nicht bearbeitet werden, oder die Ethercat-Zykluszeit in Bezug auf das Minimum der internen Firmware-Zykluszeit zu schnell ist. Eine Diskrepanz im Working Counter klärt jedoch nicht direkt die Ursache für ein Problem: Sie besagt lediglich, dass ein Datagramm von einem oder mehreren der adressierten Slaves nicht korrekt weitergeleitet wurde. Dennoch ist diese Information direkter Teil der Datagrammstruktur und wird zyklussynchron empfangen. Dem Master steht die Information somit direkt zur Verfügung, ungültige Daten werden nicht an die Applikation weiter gereicht. Die tatsächliche Fehlerursache kann mithilfe weiterer Maßnahmen über Registerzugriffe vom Master geprüft werden. Als physikalisches Feld innerhalb der Datagramme kann die Working Counter-Information auch direkt in den Aufzeichnungen von Netzwerkanalyse-Tools, wie etwa Wireshark Scans, angezeigt werden (Bild 7). Eine Diskrepanz des Working Counters ist vor allem für Datagramme, die Prozessdaten transportieren, kritisch, da er die Konsistenz der Daten widerspiegelt. Der Master kann unverzüglich und angepasst auf die spezielle Applikation reagieren und so beispielsweise mechanischen Schaden, den Ausschuss wertvoller oder potentiell gefährlicher Materialien oder irreführende Informationen an die Nutzerschnittstelle verhindern. Im Fall weniger zeitkritischer azyklischer Kommandos vermittelt eine Working Counter-Diskrepanz dem Master, dass die Parameterdaten nicht ausgetauscht wurden und das Kommando erneut gesendet werden muss.

Beispiel 1: Antriebssysteme

Antriebs-Applikationen bestehen für gewöhnlich aus mehreren Achsen, welche untereinander synchron arbeiten, wie bei dreidimensionaler Interpolation. Auf Hardware-Ebene korrespondiert jede Achse zu einem Motorantrieb mit Ethercat-Schnittstelle. Da jeder Antrieb so-

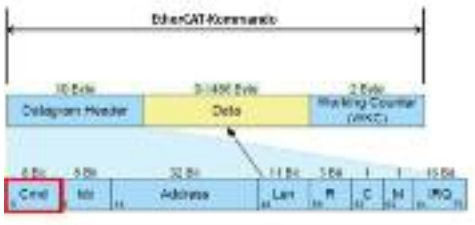


Bild 5: Kommando-Code im Datagramm

Bild 6: Working Counter-Feld am Ende des Datagramms

Bild 7: Working Counter in einer Wireshark-Aufzeichnung

Bilder: Ethercat Technology Group

Bild 8: Zyklussynchrone Reaktion der Master-Applikation

Bild 9: Zyklussynchrone Erkennung von Synchronisierungsfehlern

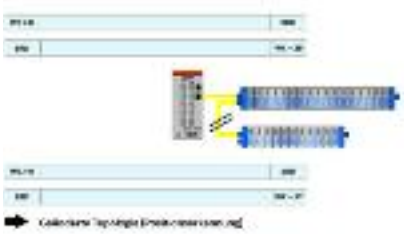


Bild 10: Zyklussynchrone Erkennung einer Topologieänderung

Bilder: Ethercat Technology Group

wohl Ausgangs- als auch Eingangsdaten austauscht (Steuerwort, Positions- und Sollwertdrehzahl, Statuswort, Positions-Istwert), werden logische Lese-Schreib-Befehle verwendet. Für den Austausch von Prozessdaten auf allen drei Antrieben wird normalerweise ein einziges Datagramm verwendet. Tritt in einem der Antriebe ein interner Hardware-Fehler, wie ein Kurzschluss in der Leistungsstufe, auf, verlässt der Slave den Status 'Operational' und unterbricht die Verarbeitung der zyklischen Daten. Dadurch erkennt die Antriebs-Applikation direkt, dass der zu dem Datagramm gehörige Working Counter nicht mehr mit dem korrekten Wert empfangen wird und kann auch die anderen zum System gehörigen Achsen stoppen (siehe Bild 8).

Beispiel 2: Synchronitätsverlust

In Applikationen, die ein hohes Maß an Synchronität der Ethercat Slaves untereinander erfordern, wird der Mechanismus der Distributed Clocks (DC), der verteilten Uhren, verwendet: Hier sind alle Slave-Uhren mit derselben Zeitreferenz abgeglichen und generieren so synchrone Ereignisse innerhalb eines jeden Slaves. Für einen korrekten Ablauf sollte das synchrone Ereignis, welches ein Prozessdaten-Update auslöst (= 'SM-Ereignis' in Bild 9), zeitlich immer nach dem Empfang des Frames, der neue Datenwerte transportiert, folgen. Da jedoch der Frame bei der Übertragung entsprechend der Master-Implementierung jittern kann, ist es möglich, dass der Frame den Slave erst nach dem internen synchronen Ereignis erreicht. Jedes Mal, wenn dieser Fall eintritt, wird ein Slave-interner Fehlerzähler erhöht. Übersteigt dieser Zähler eine vordefinierte Grenze, wechselt der Slave zurück in den Status 'SafeOperational'. Das PLC-Programm erkennt unverzüglich dieses Ereignis, da der Slave das Schreib- und Lesekommando in diesem Zustand nicht mehr verarbeitet und der Working Counter dadurch nicht erhöht wird, und kann den Synchronitätsverlust innerhalb der Software bestmöglich für die spezielle Applikation verarbeiten. Der Fehlergrund wird vom Slave in einem definierten Register ebenfalls zur Verfügung gestellt, sodass der Master dem Anwender hilfreiche Informationen zur Fehlerbehebung mitteilen kann.

Beispiel 3: Erkennung von Topologieveränderungen

Um Veränderungen in der Netzwerktopologie zyklisch zu überwachen, welche durch Hardware-Ausfälle oder die teilweise Abschaltung von Slaves verursacht werden, ist es möglich, die Working Counter-Information von Broadcast-Datagrammen zu nutzen. Ein Broadcast-Read-Kommando auf das AL Status Register wird von allen Slaves verarbeitet. Der erwartete Wert des Working Counters ist also immer direkt proportional zur Anzahl der konfigurierten Slaves. Wird ein Teil des Netzwerks ab- oder ausgeschaltet, können nicht mehr alle konfigurierten Slaves die Frames verarbeiten, wodurch sich der empfangene Working Counter-Wert der Broadcast-Datagramme entsprechend verkleinert. Über die asynchrone Abfrage von Status-Register in jedem Slave kann der Master anschließend präzise erkennen, wo sich die Topologie verändert hat. Bild 10 zeigt das Working Counter Feld vor und nach der Topologieänderung.

Fazit

Der Ethercat Working Counter ist ein einfacher und effektiver Mechanismus, welcher von Master-Applikationen genutzt werden kann, um den Netzwerkstatus synchron mit dem Kommunikationszyklus zu überwachen. Durch ihn können Master-Applikationen direkt auf anomales Verhalten innerhalb des Netzwerks reagieren. Durch den Zugriff auf die Standarddiagnoseregister, welche in jedem Slave zur Verfügung stehen, wird die präzise Erkennung von speziellen Fehlern möglich. Die genaue Beschreibung und Nutzung dieser Register erfolgt im zweiten Teil dieses Artikels. ■

www.ethercat.org

Autor: Dipl.-Ing. Florian Häfele arbeitet bei der ETG im Bereich Technologie-Support und Test

Autor: Alessandro Figini arbeitet bei der ETG im Bereich Technologie-Support und Test