EtherCAT: Studie en uitwerking didactische opstelling

Pieterjan Behaeghel

Promotor: Henk Capoen

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektrotechniek

Vakgroep Industrieel Systeem- en Productontwerp
Voorzitter: prof. Kurt Stockman
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2014-2015

UNIVERSITEIT
GENT

EtherCAT: Studie en uitwerking didactische opstelling

Pieterjan Behaeghel

Promotor: Henk Capoen

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektrotechniek

Vakgroep Industrieel Systeem- en Productontwerp
Voorzitter: prof. Kurt Stockman
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2014-2015

UNIVERSITEIT
GENT

Voorwoord

Eerst en vooral wil ik enkele mensen bedanken, zonder wie dit werk niet mogelijk zou geweest zijn. Niet in het minst wil ik mijn ouders bedanken, zonder wie ik deze studies nooit had begonnen, laat staan tot een goed einde te brengen. Daarnaast wil ik mijn beide promotoren, Henk Capoen en Thibaut Demasure, bedanken voor ondersteuning en inbreng. Tenslotte zou ik mijn vrienden willen bedanken voor de nodige afleiding tussendoor en voor het nalezen van dit werk.

Tijdens het schrijven werden bepaalde Engelse woorden en terminologieën bewust behouden omdat er geen goede Nederlandse vertaling voor bestaat. Niettegenstaande heb ik geprobeerd om deze zo goed mogelijk toe te lichten. Ik hoop dat u als lezer zich een goed beeld kan vormen over de werking van het protocol EtherCAT en al zijn mogelijkheden.

Abstract

In een wereld waar Ethernet de standaard voor LAN geworden is, kan de industrie niet achterblijven. Kan Ethernet de klassieke veldbussen vervangen? In eerste instantie blijkt dit niet mogelijk: Ethernet heeft geen realtime mogelijkheden, een cruciaal punt om toegepast te worden op procesniveau. De standaard moet dus aangepast worden om realtime werking mogelijk te maken. EtherCAT biedt zich aan als een Ethernet gebaseerde veldbus. De vraag is hoe EtherCAT van Ethernet een realtime veldbus maakt.

Het eerste probleem is de busarbitrage van Ethernet, Carrier Sense with Multiple Access and Collision Detection (CSMA/CD). Hierbij heeft iedere deelnemer gelijktijdig toegang tot de bus, wat leidt tot botsingen en onvoorspelbare back-off tijden. Om dit op te lossen hanteert EtherCAT het master-slave principe. Elk EtherCAT segment heeft 1 master die volledige controle heeft over het ganse segment. Dit impliceert echter dat slaves aangepaste software nodig hebben zodat ze niet uit zichzelf berichten op de bus kunnen plaatsen.

Om de bandbreedte van Ethernet optimaal te benutten gebruikt EtherCAT het one-total-frame principe in combinatie met Logical Addressing. De master plaatst zoveel mogelijk gegevens in éénzelfde frame en verstuurt dit doorheen het ganse netwerk. Slaves lezen en schrijven data on-the-fly in en uit het frame. Aangezien standaard Ethernet hardware dit niet mogelijk maakt moeten EtherCAT slaves aangepaste hardware hebben.

Het uiteindelijke resultaat is een veldbus met een zeer korte buscyclus. In combinatie met het gebruik van Distributed Clocks kan EtherCAT een zeer nauwkeurige synchronisatie doorvoeren. Hierdoor is deze veldbus erg geschikt voor machinebouwers waarbij nauwkeurige aansturing van uitgangen belangrijk is.

Extended abstract

XiaK is the knowledge centre for industrial automation in Kortrijk and is part of Ghent University. Companies can go there for advice and training. Therefore there is need for knowledge about real-time Ethernet protocols.

In a world where Ethernet is the standard for Local Area Networks (LAN), the question can be imposed which part Ethernet can play in the industry. Ethernet is already the standard at management level, but at process level traditional fieldbuses are still dominating. This is because Ethernet comes short of real-time possibilities.

First of all, there's the problem of media access control. Wired Ethernet uses standard Carrier Sense with Multiple Access and Collision Detection (CSMA/CD). Because every participant has simultaneous access to the network, frames can collide. This results in a random back-off time, which makes real-time operation impossible. Besides that, Ethernet is not made to transmit small amounts of data: a minimum of 46 bytes is required while most slaves have only a few bits to send.

The conclusion is that Ethernet is not made to be used as a fieldbus. Therefore, several manufacturers have developed an Ethernet-based fieldbus. The Ethernet standard is used but adapted in a way that makes real-time operation possible. For example there is PROFINET from Siemens and Phoenix Contact and EtherCAT from Beckhoff.

The first part of this master's thesis consists of a theoretical study on the EtherCAT protocol. The first question is how EtherCAT makes a real-time fieldbus based on Ethernet. Then, the special properties of the bus are regarded. The final goal is to make a summary about EtherCAT that can be used in courses. The second part of the master's thesis is to build a demo application to demonstrate the different aspects of the protocol. In the end, students will have to be able to program the application themselves in order to get acquainted with the practical side of EtherCAT.

To make a real-time fieldbus out of Ethernet, EtherCAT relies on two principles: master-slave and one-total-frame. By using the master-slave principle, the problem of media access control is solved. There's one master, an industrial PC (IPC), who has control over the entire network. Slaves cannot send messages themselves and therefore frames cannot collide.

The one-total-frame principle is applied to solve the efficiency problem. By sending all data in one Ethernet frame, overhead data only occurs once to address the entire network. Besides that, the one-total-frame principle removes the need for each slave to deconstruct and rebuild the frames, and enables each slave to read and write 'on-the-fly'. This implies that each slave has a customized chip on the datalink layer, whereas the standard Ethernet controller does not support this feature. This chip is called the EtherCAT Slave Controller (ESC) and enables EtherCAT to implement some specials features to distinguish itself from competing fieldbuses.

One of the most important features EtherCAT provides is Logical Addressing to transmit process data. With Logical Addressing, the master will transmit a part of its logical process image in one EtherCAT datagram. Multiple datagrams can be send in the same Ethernet frame. The datagram header contains information on which part of the process image is being send. When the master configures each slave, he informs them about their place in the logical process image. Together with 'on-the-fly' reading and writing of data, Logical Addressing enables a very short I/O cycle.

Another feature of EtherCAT is the possibility to accurately synchronize slaves in the network. This is made pos-

sible by the Distributed Clocks (DC) function of the ESC. Each slave has its own clock which is synchronized with a reference clock. To synchronize these clocks EtherCAT makes use of the Precision Time Protocol (PTP), as described in the IEEE 1588 standard. As a result, the clocks are synchronized with an accuracy of 100 ns.

A last important feature of EtherCAT is the support of several protocols on the application layer. These protocols are packed in a Mailbox message with a specific header to be transported on an EtherCAT network. Protocols like TCP/IP and CANopen are supported. This way, an EtherCAT network can be linked to a standard Ethernet TCP/IP network. However, a special switch that supports EtherCAT is required. By making the distinction between process data and parameter data, CANopen is a popular protocol for more complex slaves, e.g. drives. Thanks to CANopen over EtherCAT (CoE), these slaves can be integrated into EtherCAT if they use the ESC on the datalink layer.

To exemplify the practical side of EtherCAT, a demo-application has been built. The setup consists of an IPC with several I/O-modules, two brushless DC motors with accompanying encoders and drives and two remote I/O modules. With the use of a guide, the user gets acquainted with a number of features such as HotConnect, redundancy and Numerical Control.

The final goal is to synchronize the two motors in the setup by making a PLC application. To demonstrate the synchronization, both motors have been fitted with a disc with a hole pattern. A laser is directed on both discs, creating an optical effect as the motors are running synchronized. This way, the application is not breakable by means of programming errors.

EtherCAT has succeeded in creating a fieldbus based on Ethernet. By using the one-total-frame principle in combination with the high speed of Ethernet, EtherCAT achieves very high response times. The Distributed Clocks make sure the network is synchronized. However, by using the master-slave principle in combination with the custom EtherCAT Slave Controller, EtherCAT is not entirely compatible with standard Ethernet networks. Special hardware is required which comes with an extra cost.

Samenvatting

XiaK is het kenniscentrum voor industriële automatisering in Kortrijk en maakt deel uit van de Universiteit Gent. Bedrijven kunnen er terecht voor advies en opleidingen. In dat kader is er nood aan kennis over realtime Ethernet protocollen.

In een wereld waar Ethernet de standaard voor LAN-netwerken geworden is, kan de vraag gesteld worden welke rol Ethernet in de industrie kan spelen. Op managementniveau is Ethernet reeds ingeburgerd maar op procesniveau heersen de klassieke veldbussen nog steeds. Het ontbreekt Ethernet namelijk aan realtime mogelijkheden om toegepast te worden in tijdkritische toepassingen.

Eerst en vooral is er het probleem van de busarbitrage. Bij bedraad Ethernet is dit standaard Carrier Sense with Multiple Access and Collision Detection (CSMA/CD). Doordat iedereen gelijktijdig toegang heeft tot het netwerk kunnen frames botsen. Dit resulteert in een willekeurige back-off tijd waardoor realtime werking verloren gaat. Daarnaast is Ethernet niet geschikt om kleine hoeveelheden data te versturen. Het dataveld moet minimaal 46 bytes groot zijn terwijl de meeste sensoren en actoren slechts enkele bits data hebben.

Er kan dus geconcludeerd worden dat Ethernet op zich niet geschikt is om toegepast te worden als veldbus. Verschillende fabrikanten hebben daarom een Ethernet gebaseerde veldbus op de markt gebracht. Hierbij wordt een aangepaste vorm van de Ethernet standaard gebruikt om realtime werking mogelijk te maken. Zo is er PROFINET van Siemens en Phoenix Contact en EtherCAT van Beckhoff.

Het eerste deel van deze masterproef bestaat uit een theoretische studie van het EtherCAT protocol. De eerste vraag die gesteld wordt is hoe EtherCAT van Ethernet een realtime veldbus maakt. Daarnaast worden ook de speciale eigenschappen van het protocol bestudeerd. Het is uiteindelijk de bedoeling om een theoretische cursus over EtherCAT te maken. Het tweede deel van de masterproef is het bouwen van een demo applicatie die de verschillende aspecten van het protocol aantoont. Het is de bedoeling dat studenten de opstelling later zelf configureren en programmeren om zo de praktische implementatie van EtherCAT te leren kennen.

Om van Ethernet een realtime veldbus te maken baseert EtherCAT zich op twee principes: het master-slave principe en het one-total-frame principe. Door gebruik te maken van het master-slave principe wordt het probleem van de busarbitrage opgelost. Er is 1 master, een industriële PC (IPC), en die heeft controle over het ganse netwerk. Slaves kunnen uit zichzelf geen berichten op de bus plaatsen waardoor er geen botsingen kunnen voorkomen.

Daarnaast maakt EtherCAT gebruik van het one-total-frame principe om het efficiëntie probleem op te lossen. Door alle data in 1 Ethernet frame te plaatsen is er slechts 1 keer overhead data om het ganse netwerk aan te spreken. Daarnaast zorgt het one-total-frame principe ervoor dat slaves de berichten niet meer afbreken en terug opbouwen, maar dat ze on-the-fly lezen en schrijven in het Ethernet frame. Dit impliceert dat alle EtherCAT slaves een aangepaste datalinklaag moeten hebben aangezien de Ethernet standaard dit niet mogelijk maakt. De chip die dit mogelijk maakt wordt de EtherCAT Slave Controller (ESC) genoemd en bevat naast de mogelijkheid tot on-the-fly verwerken van frames ook enkele speciale functies die EtherCAT moet onderscheiden van de concurrerende veldbussen.

Eén van de belangrijkste functies die EtherCAT voorziet is Logical Addressing om procesdata te versturen. Hierbij zal de master een deel van zijn logical process image in 1 EtherCAT datagram versturen. Meerdere datagrammen worden samen in hetzelfde Ethernet frame verstuurd. In de header van het datagram wordt meegegeven welk deel van de process image verstuurd wordt. Bij configuratie van de veldbus worden slaves geïnformeerd over de

plaats van hun procesdata in de process image. In combinatie met het on-the-fly lezen en schrijven van data zorgt Logical Addressing voor een zeer korte I/O cyclus.

Daarnaast voorziet EtherCAT de mogelijkheid om slaves nauwkeurig te synchroniseren. Dit gebeurt aan de hand van de Distributed Clocks (DC) functie in de ESC. Hierbij krijgt elke slave in het netwerk een eigen klok die gesynchroniseerd wordt met een referentieklok. Om die verdeelde klokken in het netwerk te synchroniseren maakt EtherCAT gebruik van het Precision Time Protocol (PTP) zoals beschreven staat in de IEEE 1588 standaard. Het resultaat is dat de lokale klokken in het netwerk tot op 100 ns nauwkeurig lopen.

Een laatste belangrijke eigenschap van EtherCAT is de ondersteuning van verschillende protocollen op applicatieniveau. Deze protocollen worden verpakt in een Mailbox bericht met specifieke header om over een EtherCAT netwerk verstuurd te worden. Zo worden onder andere TCP/IP en CANopen ondersteund. Op die manier kan de koppeling gelegd worden naar een standaard Ethernet TCP/IP netwerk. Hiervoor is echter wel een speciale switch nodig die EtherCAT ondersteunt. Door het eenduidig onderscheid tussen procesdata en parameter data is CANopen een veelgebruikt protocol bij complexere slaves, zoals bijvoorbeeld drives. Dankzij CANopen over EtherCAT (CoE) kunnen deze slaves, mits aangepaste datalinklaag, rechtstreeks gebruikt worden in een EtherCAT netwerk.

Om de praktische kant van EtherCAT toe te lichten wordt een demo applicatie gebouwd. Deze bestaat uit een IPC met enkele I/O kaarten, 2 borstelloze DC motoren met encoder en bijhorende drives. Via een uitgeschreven stappenplan wordt kennisgemaakt met een aantal aspecten van EtherCAT zoals HotConnect, redundantie en Numerical Control.

Het doel is om uiteindelijk 2 motoren met elkaar te synchroniseren via een PLC programma. Om de synchronisatie aan te tonen wordt op beide motorassen een schijf met een gatenpatroon bevestigd. Als de motoren synchroon draaien wordt door de schijven in combinatie met een laser een optisch effect gecreëerd. Op die manier is de opstelling niet breekbaar door eventuele programmeerfouten.

EtherCAT is er dus in geslaagd om een performante veldbus te maken die gebaseerd is op Ethernet. Door het one-total-frame principe in combinatie met de hoge snelheid van Ethernet heeft het protocol een zeer snelle reactietijd. Hierbij zorgen de Distributed Clocks voor een zeer nauwkeurige synchronisatie van het netwerk. Echter door gebruik te maken van het master-slave principe in combinatie met aangepaste hardware is de veldbus niet geheel compatibel met bestaande Ethernet netwerken. Indien dit toch gewenst is kan dit via speciale hardware, wat een extra kost met zich meebrengt.

Inhoudsopgave

Lijst met afkortingen

A

ADO	Address Offset
ASIC	Application-Specific Integrated Circuit

C

CAN	Controller Area Network
CIM	Computer Integrated Manufacturing
CoE	CANopen over EtherCAT
CRC	Cyclic Redundancy Check
CSMA/CD	Carrier Sense with Multiple Access and Collision Detection

D

DC	Distributed Clocks
DPRAM	Dual-Ported Random-Access Memory

E

EAP	EtherCAT Automation Protocol
EEPROM	Electrically Erasable Programmable Read-Only Memory
EoE	Ethernet over EtherCAT
EPU	EtherCAT Processing Unit
ESC	EtherCAT Slave Controller
ETG	EtherCAT Technology Group

F

FCS	Frame Check Sequence
FMMU	Fieldbus Memory Management Unit
FPGA	Field-Programmable Gate Array

I

I^2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IP	Internet Protocol
IPC	Industrial Personal Computer
IRQ	Interrupt Request

L

LAN	Local Area Network
LLC	Logical Link Control
LVDS	Low Voltage Differential Signaling

M

MAC	Media Access Control
MII	Medium Independent Interface
MLT-3	Multi-Level Transmit 3
MPD	Master Processing Delay

N

NC	Numerical Control
NIC	Network Interface Card
NRZI	Non-Return-To-Zero Inverted

O

OSI	Open Systems Interconnection
OUI	Organizationally Unique Identifier

P

PDI	Process Data Interface/Physical Data Interface
PDO	Process Data Object
PoE	Power over Ethernet
PTP	Precision Time Protocol

S

SDO	Service Data Object
SFD	Start of Frame Delimiter
SII	Slave Information Interface
SPI	Serial Peripheral Interface

T

TCP	Transmission Control Protocol
TwinCAT	The Windows Control Automation Technology

U

UDP	User Datagram Protocol

V

VLAN	Virtual Local Area Network

W

WKC	Working Counter

X

XAE	eXtended Automation Engineering
XiaK	eXpertisecentrum industriële automatisering Kortrijk

Lijst van tabellen

Lijst van figuren

1 Inleiding

1.1 Bedrijfsvoorstelling

Het eXpertisecentrum voor Industriële Automatisering in Kortrijk (XiaK, Figuur 1.1) werd in 2013 opgericht door de Hogeschool West-Vlaanderen (Howest) als het kenniscentrum voor industriële automatisatie, machineveiligheid en mechatronica. Sinds de integratie van de opleidingen Industrieel Ingenieur in de universiteiten maakt XiaK deel uit van de faculteit Ingenieurswetenschappen en Architectuur van de Universiteit Gent.

Figuur 1.1: XiaK

1.2 Probleemstelling

In een wereld waar Ethernet de standaard wordt kan de industrie niet achterblijven. Ethernet is reeds de standaard bij particulieren en ook in het kantoor is het niet meer weg te denken. In de productiehal heersen momenteel echter nog de klassieke veldbussen. Kan Ethernet ook hier een rol kan spelen? EtherCAT biedt hier een oplossing voor door zich te profileren als Ethernet gebaseerde veldbus. De vraag is hoe EtherCAT van Ethernet een realtime veldbus maakt en hoe het zich onderscheidt van de concurrerende veldbussen.

1.3 Doelstellingen

Het eerste deel van deze masterproef betreft een theoretische studie van het EtherCAT protocol. Daarbij wordt gekeken hoe EtherCAT realtime functionaliteit naar Ethernet brengt. Daarnaast worden ook de speciale functies die EtherCAT aanbiedt bestudeerd. Op die manier moet de lezer een algemeen beeld krijgen van EtherCAT en zijn toepassingsmogelijkheden.

Het tweede deel van de masterproef is het bouwen van een praktische demo applicatie die de verschillende functies van EtherCAT aantoont. De bedoeling is dat studenten deze applicatie via een stappenplan zelf kunnen configureren om op die manier kennis te maken met de veldbus en zijn praktische eigenschappen. Zo zet EtherCAT zich vooral in op nauwkeurige synchronisatie, waardoor het vanzelfsprekend is dat dit aan bod komt in de proefopstelling. Hiervoor worden twee motoren met bijgeleverde drives ter beschikking gesteld. Aangezien de proef door studenten telkens opnieuw geprogrammeerd wordt, moet de opstelling robuust zijn. Aantonen van synchronisatie op basis van mechanische componenten is dus niet mogelijk, er moet dus een andere oplossing gezocht worden.

1.4 Projectaanpak

Zonder achterliggende kennis van het protocol kan geen praktische applicatie gebouwd worden. Daarom wordt eerst begonnen met de theoretische studie. De EtherCAT Technology Group (ETG) stelt verschillende papers en

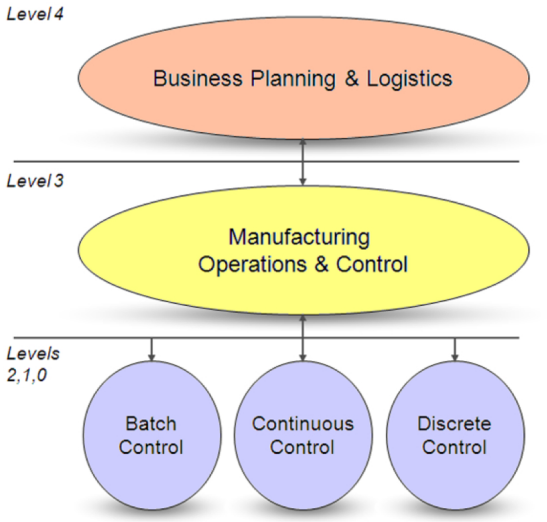
presentaties ter beschikking waarin het protocol zeer diepgaand wordt toegelicht. Daarnaast zijn de IEEE standaarden waarop de veldbus gebaseerd is vrij toegankelijk.

Van zodra er een degelijke achtergrondkennis is over de veldbus wordt begonnen met het ontwerpen van de demo applicatie. Eerst wordt het beschikbaar gestelde materiaal samen met de aangepaste software verkend. Hierbij is het de bedoeling om vertrouwd te raken met de programmeeromgeving TwinCAT. Met de verschillende mogelijkheden in het achterhoofd wordt tenslotte een demostand ontworpen waarop de beschikbaar gestelde hardware gemonteerd wordt.

2 Realtime Ethernet

2.1 Ethernet in de industrie

Ethernet is niet meer weg te denken uit de hedendaagse maatschappij. Het is de basis voor bijna alle LAN-netwerken waardoor de technologie ver ontwikkeld is. In de CIM piramide (Figuur 2.1) is Ethernet terug te vinden in lagen 2, 3 en 4 terwijl lagen 0 en 1 nog steeds ingevuld worden door de klassieke veldbussen. Het ontbreekt Ethernet echter aan realtime mogelijkheden om toegepast te worden op die onderste lagen [1].



Figuur 2.1: CIM Piramide

2.2 De vraag naar realtime Ethernet

De vraag kan natuurlijk gesteld worden waarom Ethernet dan als veldbus ingezet zou worden. Er is reeds een brede waaier aan veldbussen beschikbaar en die gelden als proven technology. En aangezien Ethernet geen realtime werking ondersteunt is er schijnbaar geen reden om de standaardisatie door te trekken naar de onderste lagen van de CIM piramide. Toch heeft Ethernet enkele eigenschappen die zeer interessant kunnen zijn om onderaan de CIM piramide toe te passen.

Een eerste eigenschap is de wijd verspreidde kennis van Ethernet. Het is het de facto standaard communicatiemiddel op lagen 2, 3 en 4 van de CIM piramide en wordt ook door particulieren in LAN-netwerken gebruikt. Het veelvuldig gebruik van Ethernet betekent ook dat de technologie ver ontwikkeld is. Dit vertaalt zich in een breed aanbod van transmissiemedia zoals twisted pair, coax, glasvezel en draadloos. Hierdoor kan Ethernet in veel omgevingen worden toegepast.

Ook de transmissiesnelheid is ongeëvenaard door een veldbus: tot 100 Gbps (IEEE 802.3bj). Dit maakt Ethernet uitermate geschikt voor netwerken met veel deelnemers of voor het verzenden van grotere berichten, zoals bijvoorbeeld parameter data. Een recente ontwikkeling is Power over Ethernet (PoE), waarmee ook voeding over de Ethernet kabel overgebracht kan worden. Alhoewel het kleine vermogens betreft (<36 W volgens IEEE 802.3af) zou dit voldoende kunnen zijn voor kleine sensoren en actoren.

2.3 Welke mogelijkheden zijn er?

Door de vele voordelen van Ethernet zijn reeds verschillende Ethernet gebaseerde protocollen op de markt. Deze kunnen ingedeeld worden in drie groepen:

- **Klasse A:** Gebruikt de volledige Ethernet TCP/IP protocolstack met een eigen applicatielaag
- **Klasse B:** Gebruikt de standaard Ethernet interface maar geen TCP/IP
- **Klasse C:** Gebruikt een aangepaste Ethernet interface en gebruikt geen TCP/IP

Een voorbeeld van een klasse A protocol is Modbus TCP. Hierbij wordt de volledige Ethernet TCP/IP stack gebruikt, waardoor alle hardware behouden kan worden. Modbus TCP heeft echter geen specifieke oplossing voor realtime werking en vertrouwt op de hoge snelheid van Ethernet. Daarnaast zijn er heel wat beperkingen op gebied van complexere slaves en is er geen synchronisatie voorzien.

Dan zijn er de klasse B protocollen waarvan PROFINET RT de bekendste is. Standaard TCP/IP wordt gebruikt om niet tijdkritische parameter data te versturen, terwijl tijdkritische gegevens deze sublaag overslaan en rechtstreeks in het Ethernet frame geplaatst worden. Hierdoor is een PROFINET RT netwerk beperkt tot één Ethernet netwerk. Om realtime werking te creëren wordt met Q-tagged frames gewerkt, waarbij het prioriteitsniveau voor realtime data het hoogst is.

Tenslotte zijn er de klasse C protocollen zoals PROFINET IRT en EtherCAT. Deze protocollen vereisen volledig aangepaste hardware, waardoor extra functies zoals synchronisatie geïmplementeerd kunnen worden. Bij PROFINET IRT wordt de netwerkcyclus opgedeeld in twee tijdsloten: één voor realtime data en één voor non-realtime data. Hiervoor zijn speciale switches nodig die dit tijdsinterval bewaken.

Naast EtherCAT is er ook het EtherCAT Automation Protocol (EAP). Dit protocol werkt op een standaard Ethernet TCP/IP netwerk met standaard hardware en wordt gebruikt voor master-master communicatie of voor SCADA en HMI toepassingen. EAP werd ontwikkeld om samen met EtherCAT de volledige CIM piramide te kunnen invullen. Met het EAP protocol is het onder andere mogelijk om variabelen uit een EtherCAT netwerk aan te spreken.

3 EtherCAT

3.1 Inleiding

EtherCAT staat voor Ethernet for Control Automation Technology en werd voor het eerst voorgesteld in 2003 door Beckhoff. Om het protocol verder te promoten werd in 2004 de EtherCAT Technology Group (ETG) opgericht. Bedrijven kunnen gratis lid worden en op die manier bijdragen aan de verdere ontwikkeling van het protocol. EtherCAT speelt vooral in op de lage kostprijs (Ethernet chips en kabels worden in massa geproduceerd), de flexibiliteit en de uitgebreide synchronisatie mogelijkheden. [2]

Figuur 3.1: EtherCAT

3.2 Algemeen Principe

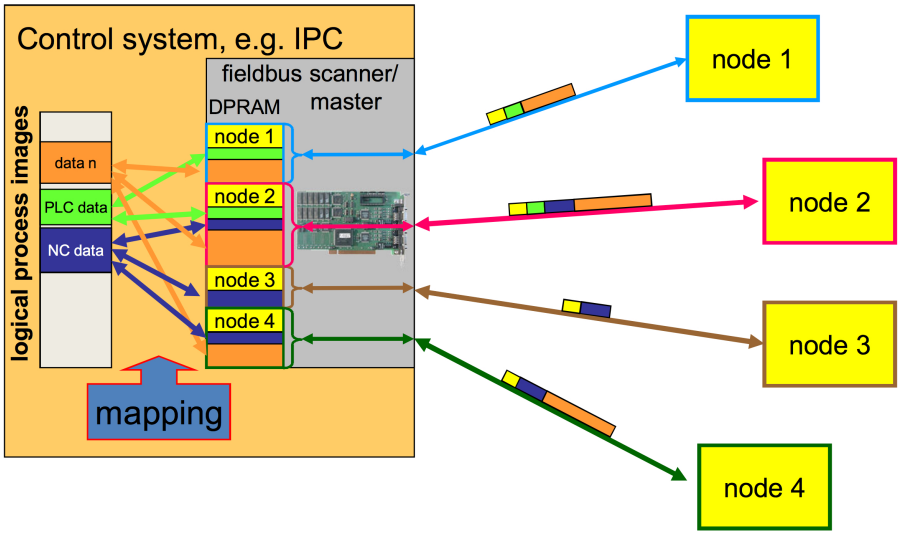
De realtime werking van EtherCAT berust op twee principes: master-slave en one-total-frame.

3.2.1 Master-slave

Om Ethernet realtime te maken moet het probleem van de busarbitrage, welke standaard CSMA/CD is, opgelost worden. Aangezien botsingen willekeurig en onvoorspelbaar gebeuren kan dit geen realtime werking garanderen. Daarnaast zorgt Carrier-Sense samen met de back-off tijd voor onvoorspelbare bustijden. EtherCAT lost dit op door gebruik te maken van het master-slave principe. Er is slechts 1 master, een IPC, en die heeft controle over het ganse netwerk. Slaves kunnen niet uit zichzelf een bericht op de bus plaatsen.

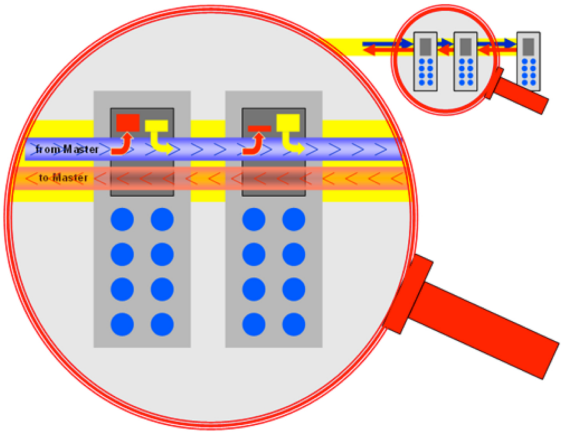
3.2.2 One-total-frame

Aangezien de master alle controle over het netwerk heeft, zijn er een aantal mogelijkheden om de data bij de slaves te krijgen en omgekeerd. Bij het afzonderlijk afpollen van elke slave moet de data uit de logical process image gehaald worden en in een apart frame verpakt worden. Dit proces heet mapping en is weergegeven in figuur 3.2. [3]



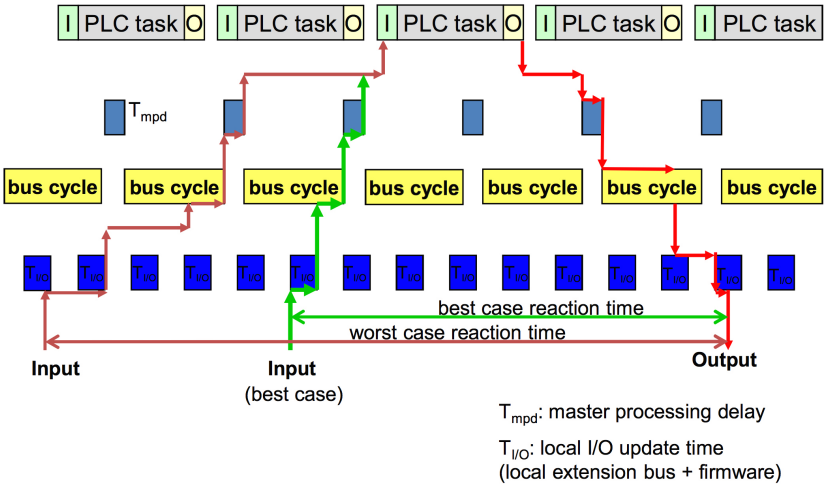
Figuur 3.2: I/O Mapping

Bij het one-total-frame principe wordt de logical process image in zijn geheel verstuurd en is er dus geen mapping nodig, met een tijdswinst T_{mpd} als gevolg. In de slaves worden de data hierdoor niet meer frame per frame verwerkt, maar on-the-fly gelezen en weggeschreven, zoals weergegeven in figuur 3.3. Dit zorgt voor een tijdswinst $T_{I/O}$.

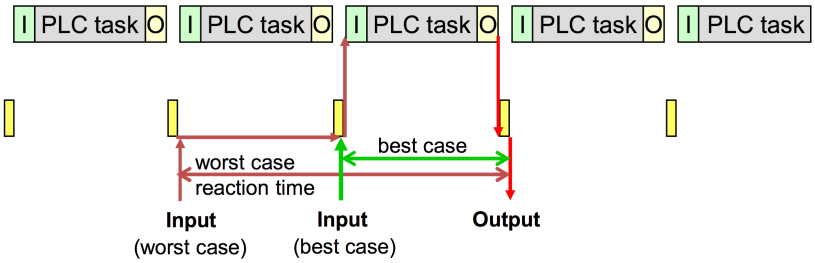


Figuur 3.3: On-the-fly lezen en schrijven

Daarnaast is er dankzij het one-total-frame principe slechts 1 keer per buscyclus overhead data. Het Ethernet dataframe moet minimum 64 bytes groot zijn, terwijl de meeste slaves slechts enkele bytes nodig hebben. Door alle data in 1 frame te versturen wordt deze inefficiëntie opgelost. In combinatie met de hoge snelheid van Ethernet resulteert dit in een zeer korte buscyclus. In figuur 3.4 is de totale I/O tijd van een traditionele veldbus weergegeven en in figuur 3.5 die van EtherCAT.



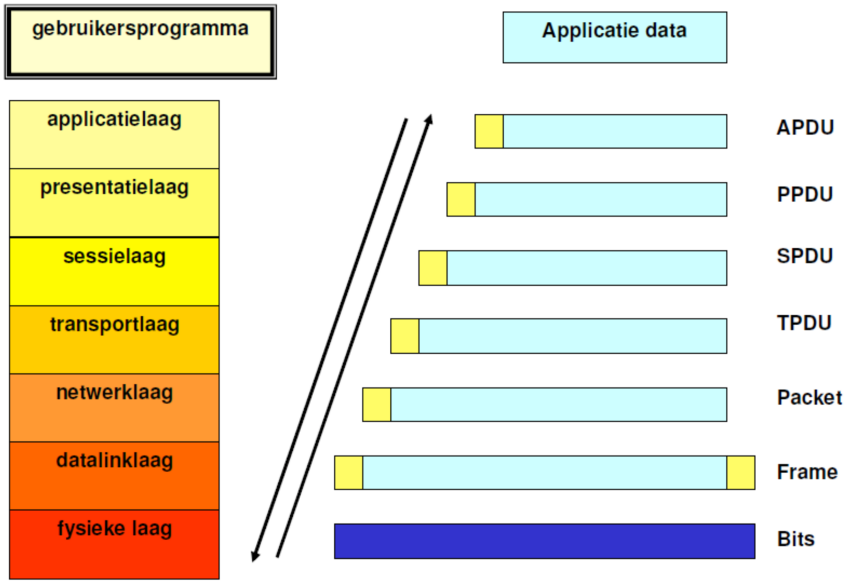
Figuur 3.4: I/O tijd standaard veldbus



Figuur 3.5: I/O tijd EtherCAT

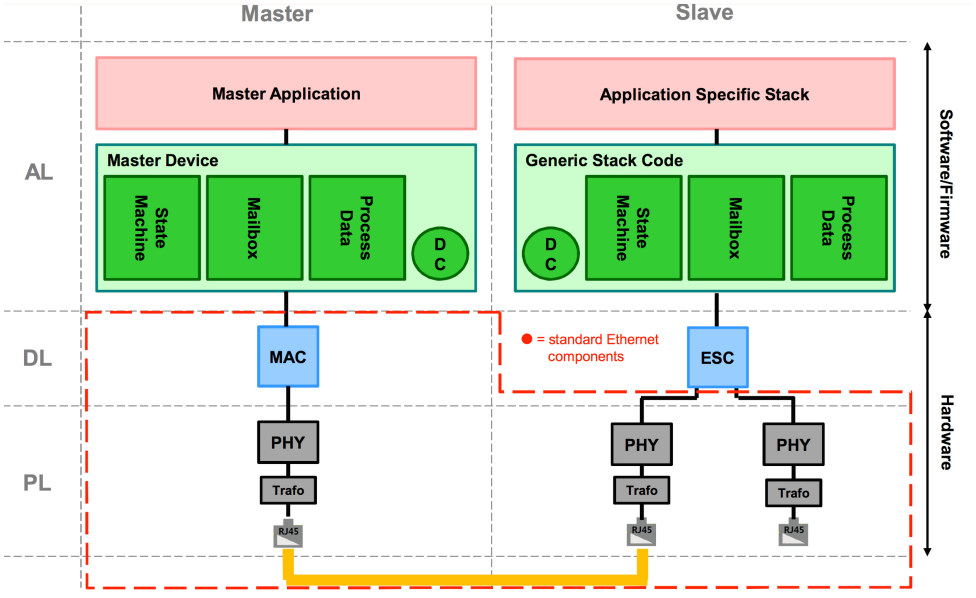
3.3 EtherCAT in het OSI-model

Wanneer data verstuurd moeten worden over een medium zijn er bijkomende gegevens nodig om deze data te kunnen versturen. Deze bijkomende gegevens worden overhead data genoemd en zijn afhankelijk van het gebruikte protocol en transmissiemedium. Deze overhead data vormen samen met de nuttige data het dataframe, waarvoor het OSI-model (Figuur 3.6) een blauwdruk is.



Figuur 3.6: OSI-model

EtherCAT maakt enkel gebruik van de fysieke laag, de datalinklaag en de applicatielaag. Daarbij moet nog een onderscheid gemaakt worden tussen de master en de slave. Zoals te zien is in figuur 3.7 heeft een EtherCAT slave een andere chip op de datalinklaag, de EtherCAT Slave Controller (ESC) genaamd. Deze zorgt voor de specifieke functies van EtherCAT in de slaves zoals het on-the-fly verwerken van de berichten.



Figuur 3.7: Protocolstack EtherCAT

3.4 Laag 1: Fysieke laag

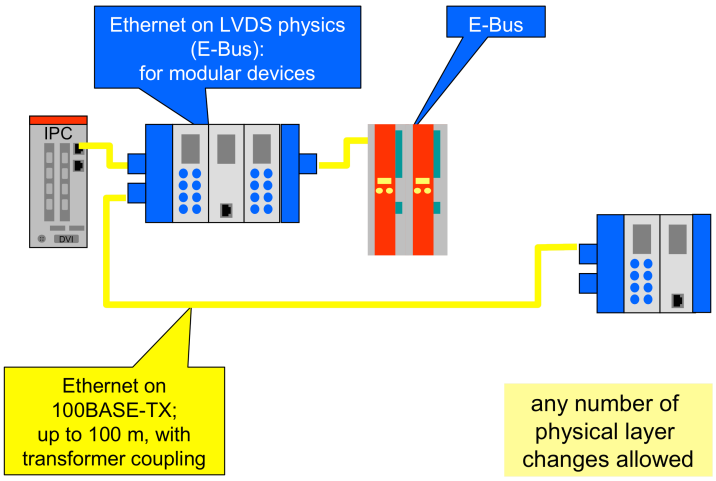
De fysieke laag wordt ingevuld door Ethernet en is beschreven in de IEEE 802.3 standaard. Deze laag behandelt de verschillende transmissiemedia en hoe de data over deze media gecodeerd worden.

3.4.1 Bekabeling

EtherCAT vereist een snelheid van 100 Mbit/s waardoor niet alle versies van de IEEE 802.3 standaard gebruikt kunnen worden. Volgende versies worden momenteel ondersteund:

- 100Base-TX: Twisted Pair, de meest gebruikte variant, vereist minimum een Cat5 kabel zonder cross-over.
- 100Base-FX: Optische vezel variant.
- LVDS: Low-Voltage Differential Signaling, een speciale variant van Ethernet die beschreven staat in de ANSI/TIA/EIA-644 standaard. Wordt gebruikt voor de E-Bus-backplane.

Binnen de EtherCAT IPC's en I/O-eilanden wordt tussen de verschillende I/O kaarten gecommuniceerd via de E-Bus-backplane. Via de buskoppelaar wordt het Ethernet signaal komende van twisted pair of optische vezel omgezet in LVDS-signaal. Dit heeft als voordeel dat er minder vertraging optreedt door het gebruik van een andere codering, die niet noodzakelijk compatibel moet zijn met de Ethernet standaard. Figuur 3.8 toont de bekabeling bij een typisch EtherCAT netwerk.

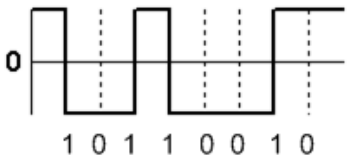


Figuur 3.8: Bekabeling typisch EtherCAT netwerk

3.4.2 Codering

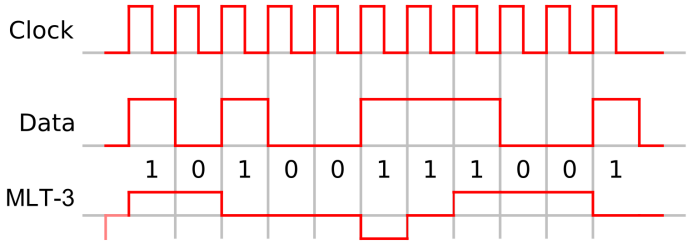
De ESC biedt zijn signalen op twee verschillende manieren aan, afhankelijk van het gebruikte transmissiemedium. Wordt de Ethernet standaard gebruikt, dan worden de gegevens aangeboden volgens de Medium Independent Interface (MII), zoals die beschreven staat in IEEE 802.3. Als de gegevens via E-Bus verstuurd worden, dan worden de gegevens volgens een eigen interface verstuurd.

Het MII-signaal wordt in de 100Base standaard van Ethernet eerst gecodeerd volgens 4B5B, waarbij elk blok van 4 bits gecodeerd wordt in een overeenkomstig blok van 5 bits. Deze codering is nodig om de daaropvolgende codering, Non-Return-to-Zero Inverted (NRZI), mogelijk te maken. Bij NRZI komt een spanningsovergang overeen met een logische '1' en geen spanningsovergang met een logische '0'. Het principe van NRZI is weergegeven in figuur 3.9. De 4B5B codering is zodanig opgebouwd dat elke 5 bits data minstens 2 spanningsovergangen hebben, wat nodig is voor synchronisatie van zender en ontvanger. NRZI-codering heeft op zich geen synchronisatie waardoor deze voorafgaande codering nodig is. Deze vorm wordt gebruikt bij de 100Base-FX (optische vezel) variant.



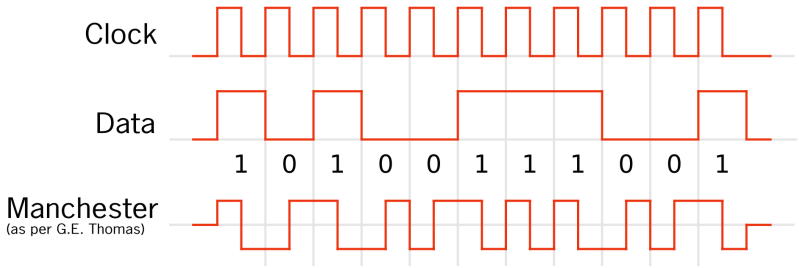
Figuur 3.9: Non-Return-to-Zero Inverted

Bij 100Base-TX (twisted pair) wordt het NRZI-gecodeerde signaal extra gecodeerd via Multi-Level Transmit 3 codering (MLT-3, Figuur 3.10). Deze vorm van codering wordt gebruikt om de fundamentele frequentie van het signaal te verlagen. Bij NRZI kan een volledige periode gerealiseerd worden met 2 bits, terwijl bij MLT-3 hiervoor minimum 4 bits nodig zijn. Dit maakt MLT-3 een ideale codering bij koperdraad als transmissiemedium.



Figuur 3.10: Multi-Level Transmit 3

Voor de E-Bus wordt een andere codering gebruikt. EtherCAT kiest hier voor Manchestercodering volgens G.E. Thomas (ook wel Bitphase L Manchestercode genoemd). Hierbij wordt elke bit opgedeeld in twee gelijke intervallen. Volgens de standaard van G.E. Thomas begint een logische '1' met een hoog signaal in het eerste interval en eindigt deze met een laag signaal. Een logische '0' wordt aangeduid door net het omgekeerde signaal: eerst laag, dan hoog. Figuur 3.11 geeft dit principe weer.



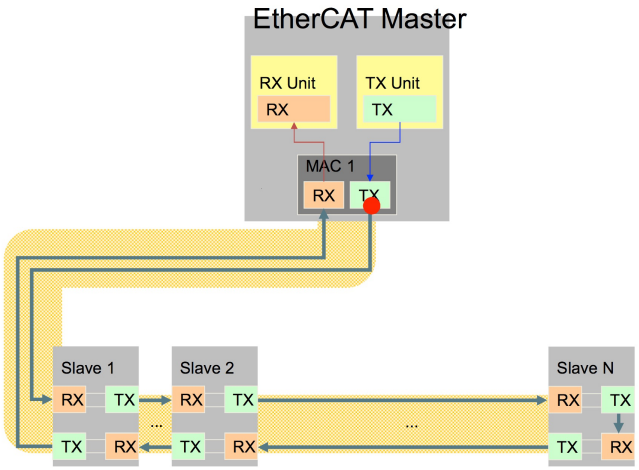
Figuur 3.11: Manchestercodering

Deze codering heeft verschillende voordelen. Ten eerste treedt er minder jitter op dan bij andere coderingen. Dit komt doordat het signaal zichzelf synchroniseert: elke bit heeft een signaalovergang. Het signaal heeft eveneens geen DC-offset, waardoor het goed bestand is tegen externe storingen. Bij Manchestercodering moet de bandbreedte van het signaal echter dubbel zo groot zijn dan andere coderingen om dezelfde bitrate te halen. EtherCAT voorziet voor de E-Bus dezelfde effectieve bitrate als de 100Base standaard: 100 Mbit/s.

Als eindcodering wordt bij de E-Bus differential signaling gebruikt. Hierbij worden twee spanningslijnen gebruikt met respectievelijk een positieve en een negatieve spanning. De polariteit wordt omgewisseld, waarbij de stroomzin bepaalt welk logisch signaal verstuurd wordt.

3.4.3 Netwerkarchitectuur

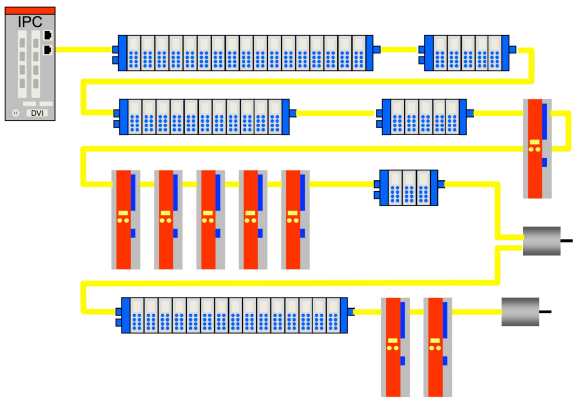
Het one-total-frame principe vereist een logische dataring, zoals te zien is in figuur 3.12. Dankzij de twisted pair kabel is dit mogelijk met één enkele kabel indien de werkingsmodus full-duplex is. Op de figuur is ook te zien dat elke EtherCAT slave minstens twee Ethernet-poorten moet hebben om het netwerk te kunnen doorverbinden.



Figuur 3.12: Logische dataring

Lijntopologie

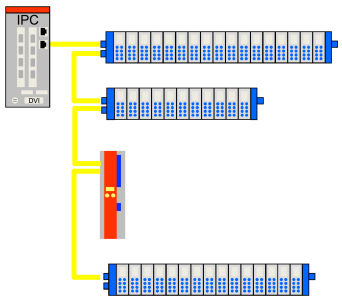
De lijntopologie (Figuur 3.13) is de meest eenvoudige, maar ook de minst redundante variant. Alle slaves worden als een ketting na elkaar geplaatst, met als gevolg dat een defecte slave alle achterliggende slaves uitschakelt.



Figuur 3.13: Lijntopologie

Busstructuur

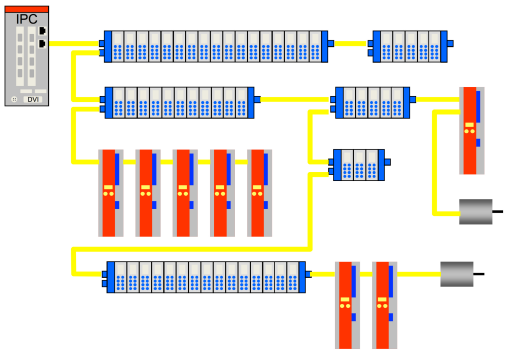
Bij de busstructuur (Figuur 3.14) wordt de aftakking naar de volgende slave niet op het einde van het E-bus segment gemaakt maar op de buskoppelaar. Op die manier kunnen defecte slaves de achterliggende deelnemers niet uitschakelen, tenzij de buskoppelaar zelf defect is.



Figuur 3.14: Busstructuur

Boomstructuur

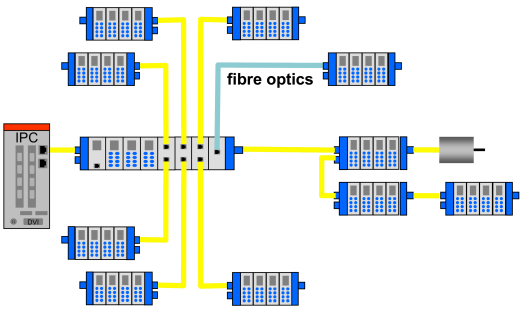
De boomstructuur (Figuur 3.15) kan gezien worden als een combinatie van de lijnstructuur en de busstructuur.



Figuur 3.15: Boomstructuur

Stertopologie

Ook een stertopologie is mogelijk, wat in sommige situaties de benodigde kabellengte kan beperken. Zoals figuur 3.16 aangeeft is het ook mogelijk om verschillende transmissiemedia door elkaar te gebruiken. Voor deze topologie zijn speciale I/O kaarten beschikbaar.

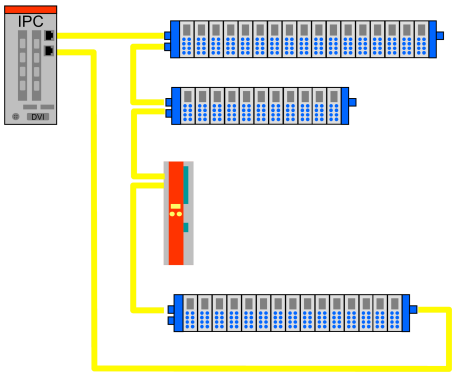


Figuur 3.16: Stertopologie

Ringstructuur

Bij voorgaande topologieën is er een beperkte redundantie mogelijk bij toestelfouten maar niet bij kabelfouten of buskopfouten. Om hieraan tegemoet te komen kan een ringstructuur opgebouwd worden. Hierbij moet de

master enkel een extra Ethernet-poort hebben, zoals te zien is op figuur 3.17. De master kan nu het dataframe in twee richtingen versturen, waardoor pas vanaf twee fouten een deel van het netwerk kan wegvallen.



Figuur 3.17: Redundantie dankzij ringstructuur

3.5 Laag 2: Datalinklaag

De datalinklaag vormt de schakel tussen de fysieke laag en de applicatielaag. Bij EtherCAT wordt een onderscheid gemaakt tussen master en slaves. De master gebruikt de Ethernet standaard terwijl de slaves een eigen EtherCAT Slave Controller (ESC) gebruiken.

3.5.1 Master

In principe heeft de master geen speciale hardware nodig om een EtherCAT netwerk aan te sturen. Aangezien de slaves uit zichzelf geen berichten op de bus kunnen plaatsen, is de bus altijd vrij voor de master en is de busarbitrage dus niet van toepassing. Een EtherCAT master kan daardoor bestaan uit een industriële PC met een standaard Ethernet NIC. In de master regelt EtherCAT alles op de applicatielaag. De datalinklaag van Ethernet bestaat uit twee deellagen: de MAC-sublaag en de LLC-sublaag.

Media Access Control sublaag

De Media Access Control sublaag maakt net zoals de fysieke laag deel uit van de Ethernet standaard (IEEE 802.3). De functie van deze laag is het opbouwen van het Ethernet frame en dit doorgeven aan de fysieke laag voor verzending. Bij het ontvangen van frames wordt de header afgebroken en de nuttige informatie doorgegeven aan de bovenliggende laag. Doordat slaves data on-the-fly lezen en schrijven, breken zij geen frames af en kunnen ze deze sublaag dus niet gebruiken. De meeste slaves zullen dan ook de data in de Ethernet header niet interpreteren, op het EtherType veld na. De MAC-sublaag moet aanwezig zijn bij de master om andere protocollen dan EtherCAT te kunnen gebruiken, zoals bijvoorbeeld TCP/IP. [4]

Daarnaast heeft de MAC-sublaag nog enkele andere taken:

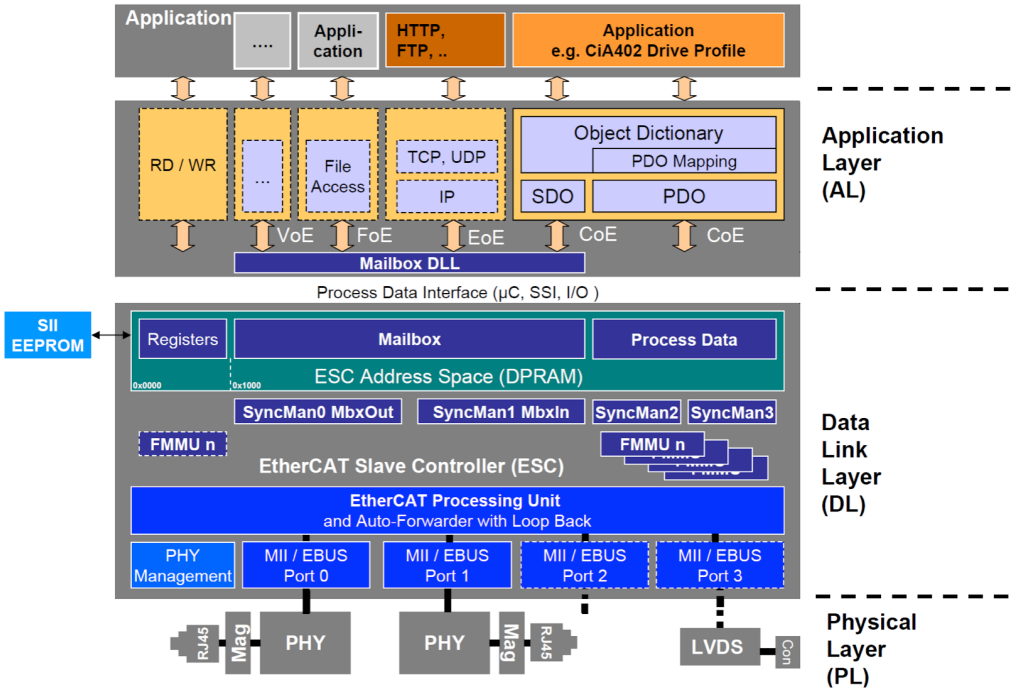
- **Busarbitrage:** Standaard is dit CSMA/CD maar aangezien de master alleen toegang heeft tot de bus vormt dit geen probleem. Vanuit dit standpunt kunnen er dan ook geen standaard switches in een EtherCAT netwerk gebruikt worden. Ook draadloos is hierdoor onmogelijk.
- **Adressering:** Via het MAC-adres wordt de hardware geadresseerd. EtherCAT heeft dit niet nodig om te werken en een slave heeft geen MAC-adres nodig. Toch kunnen MAC-adressen gebruikt worden om bijvoorbeeld het EtherCAT netwerk op te delen in verschillende segmenten.
- **Foutcontrole:** Ethernet voorziet een Frame Check Sequence (FCS) om de data integriteit te controleren.

Logical Link Control sublaag

De Logical Link Control sublaag staat beschreven in de IEEE 802.2 standaard en behoort dus niet tot de Ethernet standaard maar wordt er altijd mee in combinatie gebruikt. Deze laag dient als uniforme interface naar de onderliggende MAC-sublaag, welke afhankelijk is van het gebruikte transmissiemedium (IEEE 802.3 (Ethernet), IEEE 802.11 (WLAN), IEEE 802.5 (token ring) etc.). De LLC-sublaag geeft aan welk protocol op laag 3 of hoger gebruikt wordt. Bij EtherCAT wordt het EtherType veld van de MAC-sublaag gebruikt om aan te geven dat het om EtherCAT gaat, waardoor de LLC-sublaag overbodig is en dus niet gebruikt wordt. Deze sublaag moet echter wel aanwezig zijn om andere Ethernet gebaseerde protocollen te kunnen ontvangen. [5]

3.5.2 Slaves

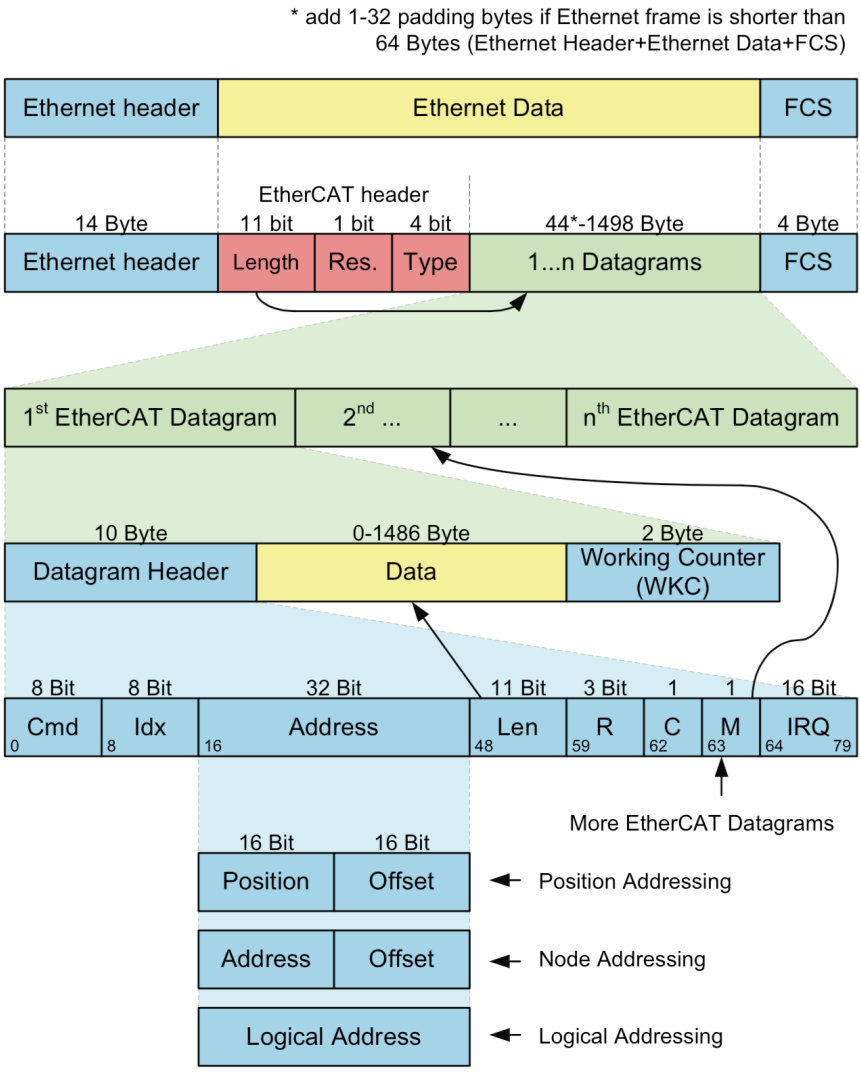
Omdat het met de standaard MAC-sublaag niet mogelijk is om data on-the-fly te lezen en te schrijven hebben de slaves een speciale EtherCAT Slave Controller. Hierin worden alle basisfuncties die een EtherCAT slave moet bezitten geprogrammeerd, aangevuld met specifieke eigenschappen naar gelang de slave. Figuur 3.18 toont de algemene structuur van een ESC binnen het OSI-model.



Figuur 3.18: EtherCAT Slave Controller binnen het OSI-model

3.6 EtherCAT dataframes

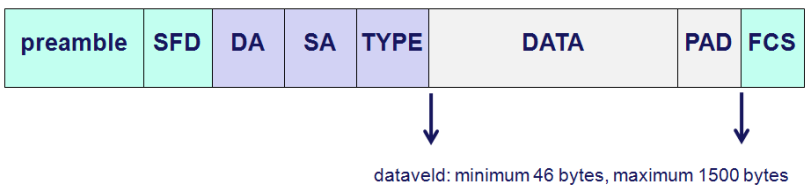
EtherCAT deelt gegevens op in één of meerdere datagrammen die worden verpakt in een Ethernet frame. Deze datagrammen worden voorafgegaan door een EtherCAT header van 2 bytes. Binnen de datagrammen zelf bevindt zich, naast de data, nog een datagram header en een Working Counter (WKC). Figuur 3.19 geeft hiervan een schematisch overzicht weer.



Figuur 3.19: EtherCAT dataframe

3.6.1 Ethernet header

De Ethernet header wordt samen met de FCS opgebouwd door de master. Slaves zullen door het on-the-fly principe deze header niet opbouwen of afbreken. Figuur 3.20 toont de opbouw van het Ethernet frame.



Figuur 3.20: Ethernet frame

Het Ethernet frame bestaat uit volgende velden:

- **Preamble (7 bytes):** Een sequentie van 56 bits afwisselend 1 en 0 die dienen als synchronisatie en om elke deelnemer de kans te geven de busactiviteit waar te nemen.
- **SFD (1 byte):** Start of Frame Delimiter (1010 1011), geeft aan dat de preamble ten einde is en dat nuttige informatie volgt.
- **Destination Address (6 bytes):** Het MAC-adres van de bestemmeling van het bericht.
- **Source Address (6 bytes):** Het MAC-adres van de verzender van het bericht.
- **Length/Type (2 bytes):** Indien de waarde van dit veld kleiner is dan 1500 decimaal, dan geeft dit veld de lengte van het dataveld in het frame aan. Is de waarde groter dan 1536 decimaal dan staat de inhoud voor het EtherType veld, wat aangeeft welk protocol er in de dataveld gebruikt wordt. Voor EtherCAT wordt de waarde van dit veld 0x88A4.
- **Data:** De hoeveelheid nuttige data in het frame is afhankelijk van het gebruikte type frame. EtherCAT gebruikt standaard Ethernet frames, waardoor de lengte tussen 46 bytes en 1500 bytes moet liggen. Bij 46 bytes wordt door de overhead de minimum lengte van een volledig Ethernet frame (72 bytes incl. preamble en SFD) bereikt.
- **PAD:** Als de nuttige data minder zijn dan 46 bytes worden willekeurige padding bits toegevoegd om de minimale lengte te bereiken.
- **FCS (4 bytes):** Frame Check Sequence: een CRC-check van 32 bits wordt aan het einde toegevoegd als foutcontrole. Aangezien de data in het bericht door de slaves aangepast wordt moeten deze de FCS herberekenen.

De meeste EtherCAT slaves hebben geen MAC- en IP-adres en verwerken Ethernet frames met eender welk adres, enkel het EtherType veld is van belang. Alle andere frames worden ofwel vernietigd ofwel doorgestuurd naar de volgende poort, afhankelijk van de instellingen.

EtherCAT kan ook werken met IP in combinatie met UDP, wat EtherCAT over Internet genoemd wordt. De waarde van het EtherType veld wordt dan 0x0800 (IPv4) en de UDP-poort wordt 0x88A4 (EtherCAT). Ook Q-tagged frames (IEEE 802.1Q) voor VLAN's worden ondersteund, maar EtherCAT slaves negeren de informatie in de VLAN-tag. Er kunnen dus geen VLAN's met verschillende slaves opgemaakt worden.

3.6.2 EtherCAT header

De EtherCAT header bestaat uit drie delen:

- **Length (11 bits):** Lengte van alle EtherCAT datagrammen in het frame samen, uitgedrukt in bytes.
- **Reserved (1 bit):** Gereserveerde bit, altijd 0.
- **Type (4 bits):** Het soort datagram. EtherCAT slaves ondersteunen enkel EtherCAT datagrammen. Naast EtherCAT kunnen ook EAP berichten over een EtherCAT netwerk verstuurd worden. De verschillende mogelijkheden voor het Type-veld zijn weergegeven in tabel 3.1.

Tabel 3.1: Waarden EtherCAT Type Veld

Waarde (DEC)	Betekenis
0	Gereserveerd
1	EtherCAT
2 - 3	Gereserveerd
4	EAP Process Data
5	EAP Mailbox Data
6- 15	Nog niet in gebruik

EtherCAT slaves bekijken enkel het type veld van deze header. De overige informatie is voor de master bedoeld.

3.6.3 EtherCAT datagram

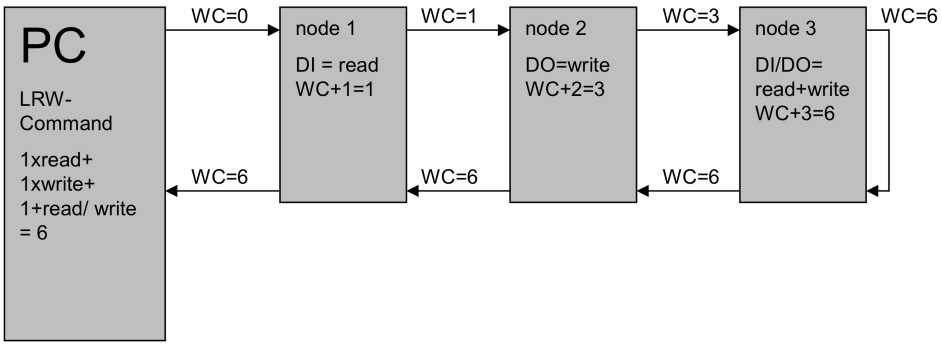
Een Ethernet frame kan maximaal 15 EtherCAT datagrammen bevatten. Zijn er meer datagrammen nodig dan wordt een tweede Ethernet frame aangemaakt. Elk EtherCAT datagram bestaat uit drie delen:

- **Datagram header (10 bytes):** Elk datagram heeft een header die informatie over de data bevat, zodat slaves de juiste acties uitvoeren.
- **Data (tot 1486 bytes):** De nuttige data binnen het datagram.
- **Working Counter (2 bytes):** De Working Counter (WKC) is een controlemechanisme van EtherCAT om eventuele lees- of schrijffouten te detecteren. Het commando in de datagram header bepaalt welke actie uitgevoerd moet worden. Afhankelijk van de uit te voeren actie wordt door de slave een waarde bijgeteld, zoals weergegeven in tabel 3.2. De master weet welke acties uitgevoerd moeten worden en weet dus welke waarde voor de WKC hij mag verwachten. Klopt deze waarde niet dan is er een fout gebeurd.

Tabel 3.2: Working Counter increments

Command	Resultaat	Increment waarde
Read Command	No succes	0
	Successful Read	+1
Write Command	No succes	0
	Successful Write	+1
Read/Write Command	No succes	0
	Successful Read	+1
	Successful Write	+2
	Sucessful Read/Write	+3

Ter verduidelijking is in figuur 3.21 een voorbeeld gegeven hoe de WKC drie keer aangepast wordt met een Read/Write Command.



Figuur 3.21: WKC voorbeeld

3.6.4 Datagram header

De datagram header bestaat uit acht delen:

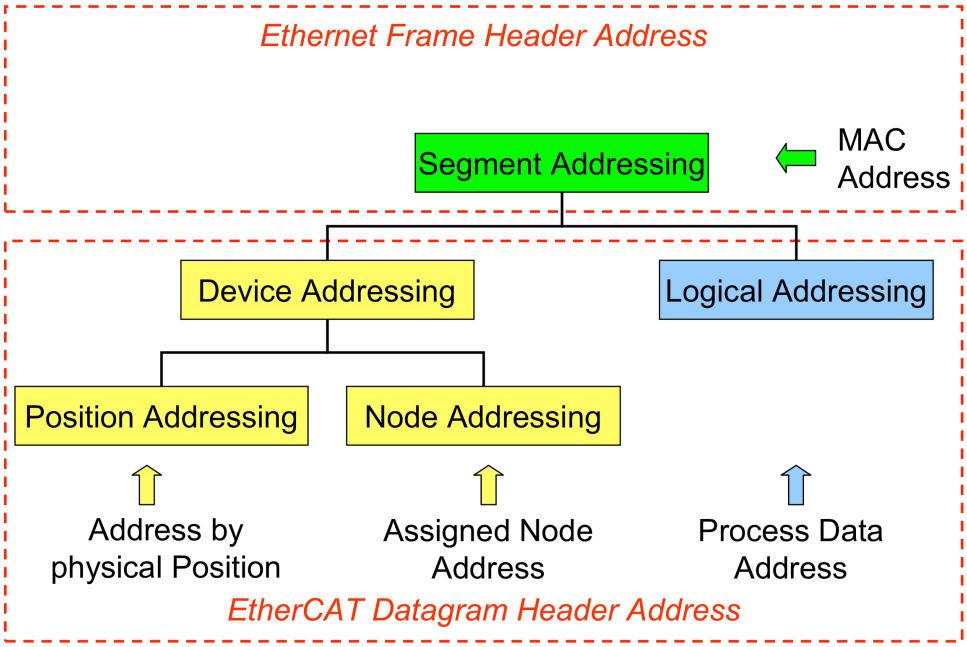
- **Cmd (8 bits):** Het EtherCAT commando geeft aan welke actie op de data uitgevoerd moet worden (lezen, schrijven of lezen en schrijven). Daarnaast wordt aan de hand van het commando een onderscheid gemaakt tussen de verschillende adresseringsvormen. Tabel 3.3 geeft een overzicht van de verschillende commando's.
- **Idx (8 bits):** De index is enkel voor de master bedoeld ter identificatie van de datagrammen. Bij een foutieve WKC kan de master aan de hand van de index het foutieve datagram opnieuw versturen.
- **Address (32 bits):** Het adres van de gegevens, wordt opgedeeld in twee delen van elk 16 bits. Hiervoor zijn drie mogelijke instellingen: Position Address, Node Address en Logical Address. Het commando geeft aan welke adressering gebruikt wordt.
- **Len (11 bits):** De lengte van de data in het datagram, uitgedrukt in bytes.
- **R (3 bits):** Gereserveerde bits, drie maal 0.
- **C (1 bit):** De circulating frame bit. Wordt door de ESC van een slave, van wie de fysieke link op poort 0 verloren gaat, op 1 geplaatst, zodat het frame vernietigd wordt en niet gaat circuleren doorheen de overgebleven slaves.
- **M (1 bit):** Geeft aan dat er nog een EtherCAT datagram volgt na dit datagram.
- **IRQ (16 bits):** Interrupt Request, wordt gebruikt door de Distributed Clocks eenheid.

Tabel 3.3: Command types

Waarde (DEC)	Afkorting	Betekenis
0	NOP	No Operation
1	APRD	Auto-increment Read
2	APWR	Auto-increment Write
3	APRW	Auto-increment Read/Write
4	FPRD	Configured Address Read
5	FPWR	Configured Address Write
6	FPRW	Configured Address Read/Write
7	BRD	Broadcast Read
8	BWR	Broadcast Write
9	BRW	Broadcast Read/Write
10	LRD	Logical Memory Read
11	LWR	Logical Memory Write
12	LRW	Logical Memory Read/Write
13	ARMW	Auto-increment Read Multiple Write
14	FRMW	Configured Read Multiple Write
15-255	/	Gereserveerd

3.7 Adressering

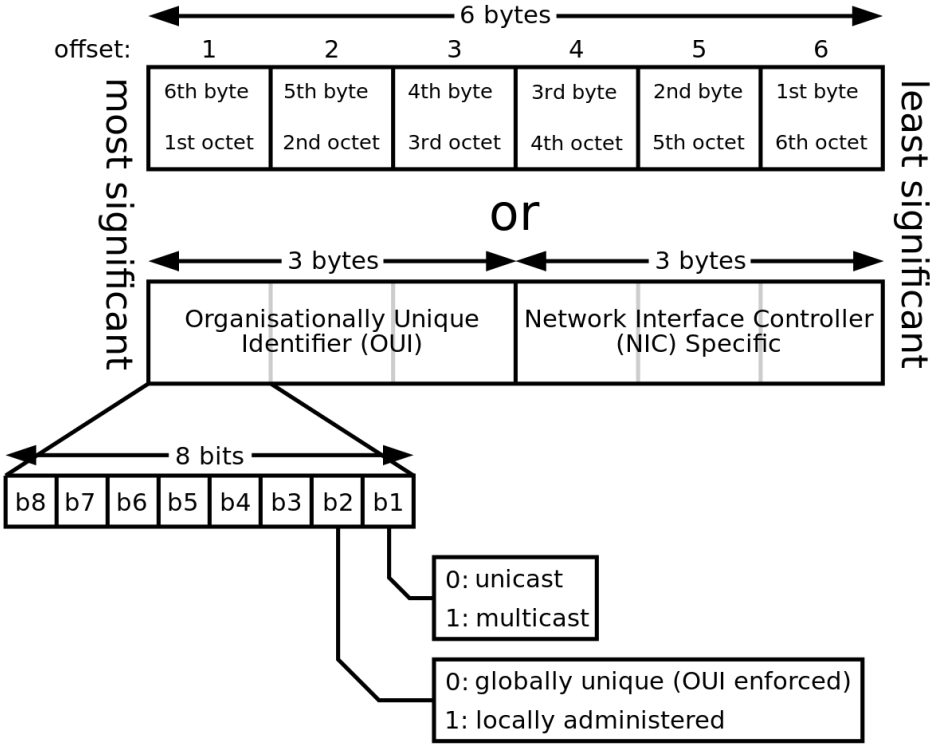
EtherCAT voorziet verschillende vormen van adressering, afhankelijk van de toepassing van de gegevens in het datagram. Figuur 3.22 geeft een schematisch overzicht weer van de verschillende adresseringsmogelijkheden.



Figuur 3.22: EtherCAT adressering

3.7.1 Segment Addressing

Segment Addressing is gebaseerd op het MAC-adres en bevindt zich dus in de Ethernet-header. Het MAC-adres is 48 bits lang en kan opgedeeld worden in twee belangrijke basisdelen, zoals te zien is op figuur 3.23.

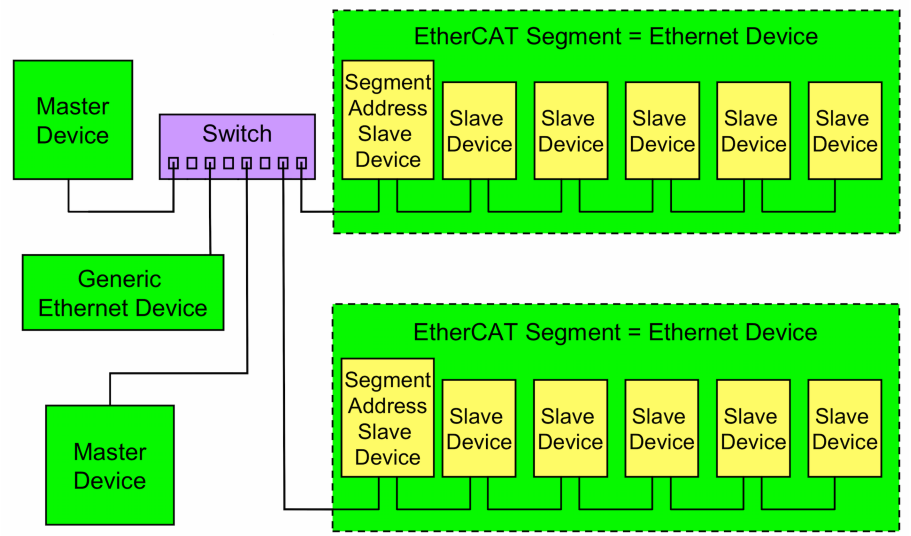


Figuur 3.23: Opbouw van een MAC-adres

Het 48-bit adres wordt opgedeeld in twee gelijke delen van elk 3 bytes. Het eerste deel is eigen aan de fabrikant en wordt de Organizationally Unique Identifier (OUI) genoemd. Het tweede deel is uniek voor elke NIC van die fabrikant. De eerste 2 bits van de OUI zijn variabel en kunnen aangepast worden, naargelang het type adressering dat gebruikt wordt.

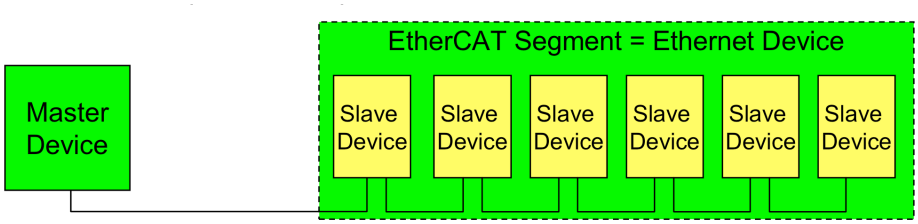
Zoals aangegeven in figuur 3.23, kan er geadresseerd worden volgens unicast of multicast. Daarnaast kan er meegegeven worden als het MAC-adres globaal uniek is of lokaal gespecificeerd wordt. Een globaal uniek adres wordt bijvoorbeeld gebruikt om één enkel toestel te adresseren volgens zijn uniek MAC-adres, terwijl lokaal beheerde adressen bepaalde groepen van toestellen kunnen zijn, die door het netwerk zelf vastgelegd zijn en herkend worden.

EtherCAT maakt van dit laatste gebruik om een onderscheid te maken tussen verschillende EtherCAT segmenten. Dit is van toepassing als het EtherCAT segment via een standaard Ethernet netwerk verbonden is met de master. Hierdoor kan de master met één enkele netwerkpoort meerdere EtherCAT segmenten beheren. De switches in het netwerk moeten speciale EtherCAT switches zijn om de realtime werking te kunnen behouden. Niet alle EtherCAT slaves hebben een MAC-adres waardoor enkel speciale slaves rechtstreeks op een Ethernet netwerk aangesloten kunnen worden. Deze vorm van Segment Addressing wordt Open Mode adressering genoemd en slaves die dit ondersteunen worden Open Mode slaves genoemd. Dit principe is weergegeven in figuur 3.24.



Figuur 3.24: Open Mode Segment Addressing

Slaves die geen Open Mode adressering ondersteunen en bijgevolg geen MAC-adres bezitten, moeten rechtstreeks met de master verbonden worden. Deze vorm van Segment Addressing wordt Direct Mode genoemd en is weergegeven in figuur 3.25.



Figuur 3.25: Direct Mode Segment Addressing

3.7.2 Device Addressing

Device Addressing gebeurt op het niveau van het EtherCAT datagram en wordt gebruikt om acyclische data te versturen. Het 32-bits adres wordt opgedeeld in twee gelijke delen van 16 bits, zoals te zien is op figuur 3.19. De betekenis van de delen is afhankelijk van het type Device Addressing dat gebruikt wordt.

Position Addressing

Bij de opstart van het systeem weet de master nog niet welke slaves aangesloten zijn en wat hun adressen zijn. Om dit op te lossen wordt Position Addressing gebruikt. Hierbij zullen de eerste 16 bits van het adres een negatieve waarde bevatten, het Position Address genoemd. Elke slave die het bericht leest verhoogt deze adreswaarde met 1. Als het Position Address op 0 komt te staan wordt de eerste slave die dit bericht ontvangt aangesproken. Ook deze slave verhoogt de adreswaarde met 1, zodat de achterliggende slaves niet meer worden aangesproken.

In praktijk zal de master beginnen met een bericht met Position Address 0 te versturen, waardoor de eerste slave wordt aangesproken. Vervolgens wordt een bericht met Position Address -1 doorgestuurd, zodat de daaropvolgende slave wordt aangesproken. Dit gaat zo verder tot er geen slave meer wordt aangesproken en dus het ganse netwerk gekend is bij de master.

In het tweede deel van het 32-bit adres wordt de offset meegegeven. Dit deel geeft aan welke geheugenplaats in de slave aangesproken wordt. Bij de opstart van het systeem zal het offset adres verwijzen naar het hardware-info register van de slave, zodat de master weet om welk type slave het gaat. Daarna zal de master op dezelfde manier aan elke slave een node-adres toekennen, zodat via Node Addressing kan worden verder gewerkt.

Een broadcast bericht is een speciale vorm van Position Addressing, waarbij de master een bericht met Position Address 0 doorstuurt. Het commando geeft aan dat het om een broadcast bericht gaat, waardoor alle slaves geadresseerd worden. Position Addressing kan problemen opleveren als bepaalde hardware onbereikbaar of defect is, maar het frame toch doorgaat. De adressering in het bericht kan dan foutief zijn waardoor verkeerde slaves aangesproken worden. Daarom gaat de voorkeur naar Node Addressing.

Node Addressing

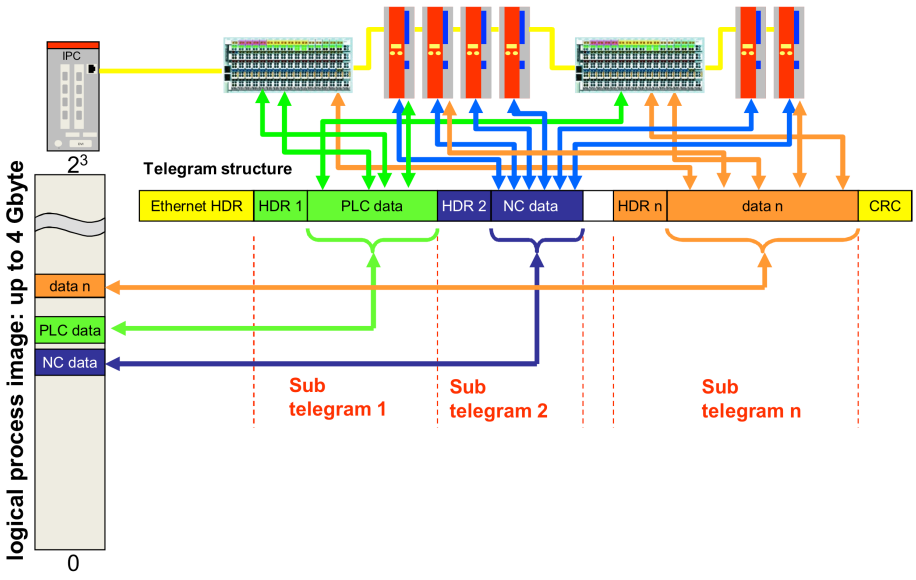
Als elke slave een Node Address gekregen heeft, wordt Node Addressing toegepast. Het eerste deel van het 32-bit adres is nu het Node Address, dat door de master wordt toegekend bij de opstart van het systeem. Indien gewenst kan de gebruiker dit adres zelf aanpassen. In totaal kan een EtherCAT segment hierdoor tot 65 535 (2^{16}) slaves bevatten.

Het tweede deel van het adres is opnieuw de offset en verwijst, net als bij Position Addressing, naar een geheugenplaats in de slave. Hierdoor kan het geheugen van een slave maximaal 64 kByte groot zijn (2^{16}).

3.7.3 Logical Addressing

Logical Addressing wordt gebruikt om procesdata efficiënt te versturen. Hierbij wordt de volledige logical process image (of een deel ervan) in een enkel datagram verstuurd. Om dit te realiseren is er nood aan speciale Functional Memory Management Unit (FMMU) functieblokken in de EtherCAT Slave Controller. Deze functieblokken zorgen voor het bitsgewijs mappen van de gegevens in het datagram naar hun overeenkomstige plaats in het geheugen van de slaves.

Aangezien bij elk EtherCAT datagram ook een commando hoort, kunnen niet alle data met hetzelfde datagram verstuurd worden. De master maakt daarom een opdeling van zijn logical process image naargelang de actie die met de data moet worden uitgevoerd (Read, Write of Read/Write). Figuur 3.26 toont dit principe.

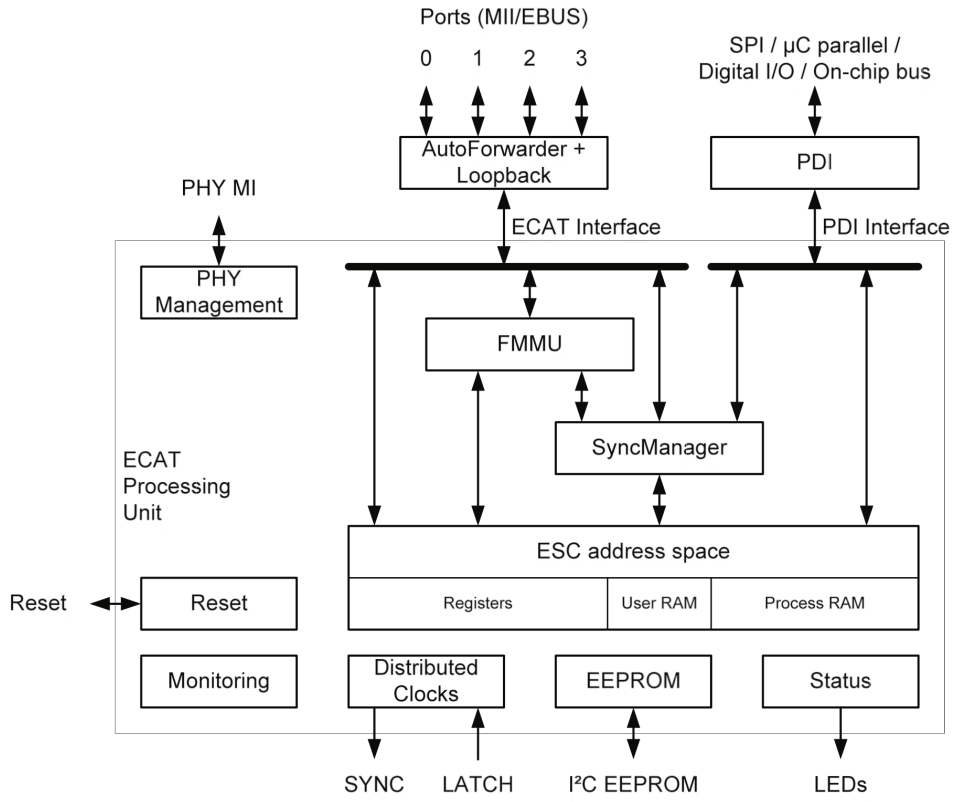


Figuur 3.26: Logical process image mapping door FMMU's

Het 32-bit adres in de datagram header verwijst nu naar het startadres van het deel van de logical process image dat in het datagram verwerkt is. In combinatie met het lengte veld weten de FMMU's welk deel van de process image in het datagram zit. De FMMU's worden tijdens de opstart door de master geïnformeerd over de positie van hun procesdata in de logical process image, waardoor deze weten op welke plaats en in welk datagram hun data zitten. Het is de taak van de FMMU's om de gegevens uit het datagram te halen en in het procesgeheugen van de slaves te plaatsen of omgekeerd.

3.8 EtherCAT Slave Controller

De EtherCAT Slave Controller vormt de basis voor elke EtherCAT slave en bevat alle basisfuncties van EtherCAT. Figuur 3.27 geeft de verschillende functieblokken en interfaces van de ESC weer.



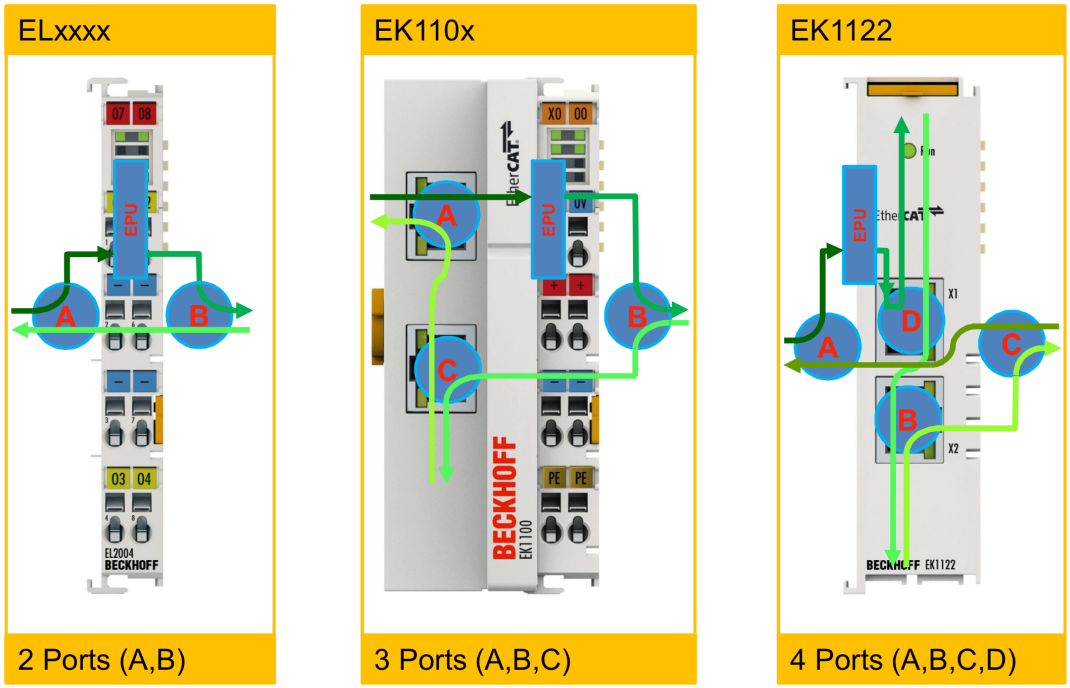
Figuur 3.27: Structuur van de EtherCAT Slave Controller

De belangrijkste functies van de ESC zijn geprogrammeerd in de EtherCAT Processing Unit (EPU):

- **EtherCAT Interfaces / Ports**
- **Process Data Interface**
- **Memory**
- **FMMU**
- **SyncManager**
- **Distributed Clocks**

3.8.1 EtherCAT interface / Poorten

De poorten (ports) van de ESC vormen de toegang naar de fysieke laag en zijn een belangrijk aspect voor het opbouwen van de logische dataring. Afhankelijk van het transmissiemedium zijn hiervoor twee interfaces voorzien: de MII en de E-Bus interface. Een slave heeft minimaal twee en maximaal vier poorten, logisch aangeduid met 0, 1, 2 of 3 en fysisch met respectievelijk de letters A, B, C of D. Poort 0 is hierbij altijd de poort die het dichtst bij de master staat en is daardoor de enige *upstream*-poort, alle andere poorten zijn *downstream*-poorten. Figuur 3.28 toont waar de poorten zich op hardware van Beckhoff bevinden.



Figuur 3.28: Fysieke poorten

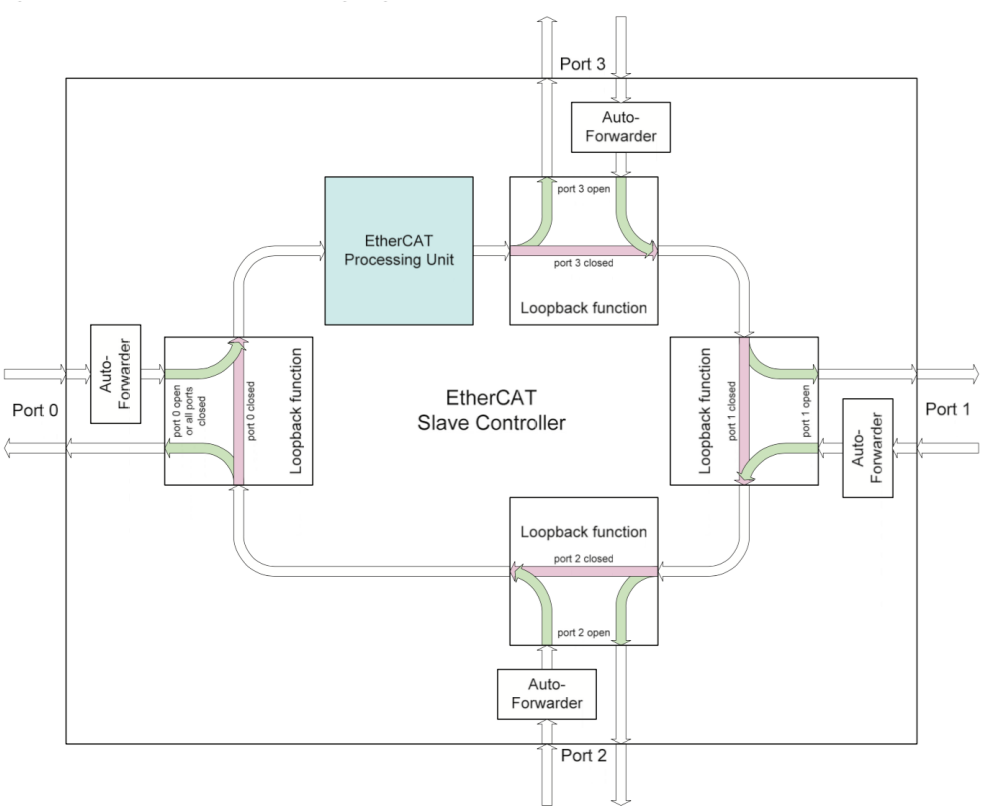
3.8.2 Loopback function en auto-forwarder

Elke poort heeft een loopback function en een auto-forwarder die zorgen voor de routing van de Ethernet frames. Deze functies behoren niet tot de EPU maar wel tot de ESC. Figuur 3.29 toont de locatie van de poorten en de EPU binnen de ESC. De auto-forwarder voert de CRC-check van de frames uit en stuurt deze vervolgens door naar de loopback function. Deze zal op zijn beurt de frames doorsturen naar de EPU of naar de eerstvolgende beschikbare poort. Via de master kan het gedrag van de loopback function van elke poort apart ingesteld worden. Er zijn vier werkingsmodi te onderscheiden:

- **Manual Open:** De poort is altijd open, onafhankelijk van de aanwezigheid van een fysische link. Bij afwezigheid van een fysieke link gaan de gegevens verloren.
- **Manual Close:** De poort is gesloten, ongeacht de status van de fysieke link. Berichten gaan niet verloren, maar alle toestellen aangesloten op deze poort zijn niet bereikbaar.
- **Auto:** De poort wordt automatisch geopend bij de aanwezigheid van een fysieke link en automatisch gesloten bij het ontbreken ervan. Dit is de standaard instelling.
- **Auto Close:** Gelijkaardig aan Auto maar met dat verschil dat bij het wegvallen van een fysieke link de poort dicht blijft, ook al wordt de link hersteld. Enkel de master kan beslissen om de poort daarna terug te openen.

Enkel poort 0 van een slave wordt bij het manueel sluiten automatisch heropend om de slave toegankelijk te houden. Daarnaast is het mogelijk dat bij het wegvallen van een fysieke link op poort 0 frames gaan circuleren

doorheen de overgebleven slaves. Deze situatie kan voorkomen indien het rondgaande frame output-data bevat. De slaves zien telkens opnieuw hetzelfde bericht binnen de verwachte cyclustijd en geven dus geen fout. Om dit tegen te gaan wordt bij het wegvallen van een link op poort 0 de circulating bit van het frame hoog geplaatst. Van zodra een slave dit frame ontvangt wordt het vernietigd in plaats van doorgestuurd.



Figuur 3.29: Loopback function en auto-forwarder binnen de ESC

3.8.3 Process Data Interface

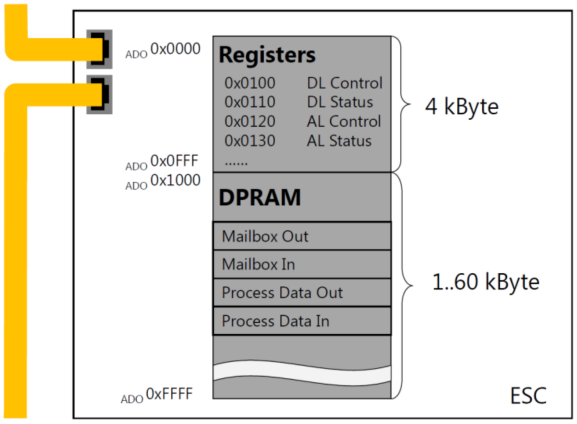
De Process Data Interface (PDI) is de toegang voor de applicatielaag naar de datalinklaag van de slave. Afhankelijk van de toepassing zijn er ESC's met verschillende interfaces mogelijk:

- **Digitale I/O:** 8 tot 32 bits aan digitale I/O, met ondersteuning voor Distributed Clocks.
- **SPI slaves:** Serial Pheripheral Interface voor slaves die seriële communicatie voorzien, zoals buskoppelaars naar andere veldbussen.
- **8/16 bit microcontroller:** Ondersteuning voor microcontrollers, nodig voor complexere slaves.
- **On-chip bus:** Specifieke bus voor FPGA's en ASIC's, afhankelijk van fabrikant tot fabrikant.
- **General Purpose I/O:** Algemene data-uitwisseling tussen applicatie en registers van de ESC, voor overige toepassingen.

3.8.4 Memory

Het geheugen van een ESC is maximaal 64 kByte groot en bestaat uit twee delen. Zoals te zien is op figuur 3.30 bevat het eerste deel (4 kByte) de registers met de actuele instellingen van de ESC, het tweede deel (tot 60 kByte)

is het Dual-Ported RAM (DPRAM) en bevat het procesgeheugen en het gebruikersgeheugen van de slave. Het geheugen is zowel toegankelijk vanuit de master als vanuit de applicatielaag van de slave.

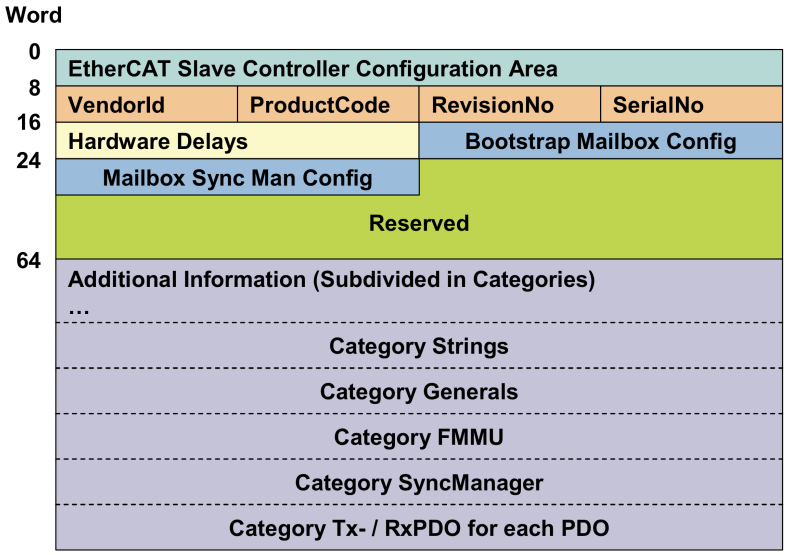


Figuur 3.30: Opbouw geheugen

De opbouw van het registerdeel van het geheugen ligt vast voor elke slave. Op die manier kan de master onafhankelijk van het type slave enkele basisconfiguraties uitvoeren, zoals bijvoorbeeld een Node Address toekennen. De grootte en invulling van het DPRAM is afhankelijk van de toepassing van de slave.

3.8.5 EEPROM

Het Electrically Erasable Programmable Read-Only Memory (EEPROM) is het niet-vluchtige geheugen van de ESC en bevat alle gegevens die bij spanningsuitval niet verloren mogen gaan, zoals configuratiegegevens en hardware-info. Sommige gegevens kunnen door de master aangepast worden, zoals bijvoorbeeld de aanmaak van SYNC signalen. Alle gegevens worden opgeslagen volgens de Slave Information Interface (SII), zodat de master de gegevens onafhankelijk van de slave kan uitlezen. Figuur 3.31 toont de opbouw van het SII EEPROM. Hierbij zijn de velden vanaf word 0 tot word 64 verplicht. De overige registers zijn afhankelijk van de complexiteit van de slave.



Figuur 3.31: Opbouw Slave Information Interface

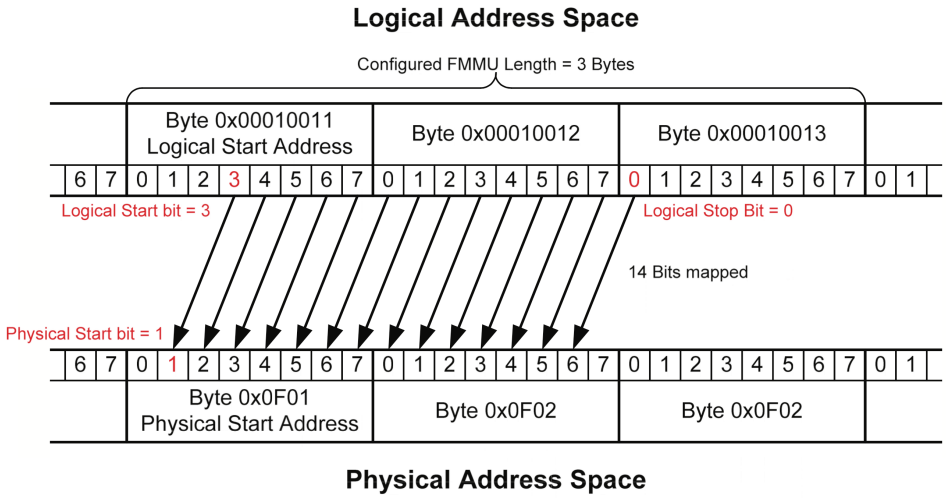
3.8.6 FMMU

De Fieldbus Memory Management Units (FFMU's) worden gebruikt bij Logical Addressing en zorgen voor de mapping van de procesdata in datagrammen naar het procesgeheugen in de ESC, zoals weergegeven in figuur 3.26. Tijdens het opstarten van het systeem zal de master aan de FMMU laten weten op welke plaats in het datagram de procesdata voor zijn slave zitten. Op die manier kan snel en on-the-fly data uitgelezen en weggeschreven worden en is er minder overhead data, wat de efficiëntie van het netwerk ten goede komt.

FMMU's worden geconfigureerd volgens hun datatype (Read, Write of Read/Write) en kunnen enkel een aaneensluitend deel van de logical process image mappen. Om meerdere niet aaneensluitende gegevens te kunnen mappen heeft een slave meerdere FMMU's nodig. Tabel 3.4 toont een voorbeeld van hoe een FMMU geconfigureerd wordt en figuur 3.32 toont het resultaat van deze instellingen.

Tabel 3.4: FMMU voorbeeldconfiguratie

FMMU configuratieregister	Registeradres	Waarde
Logical Start Address	0x0:0x3	0x00010011
Length (Bytes)	0x4:0x5	3
Logical Start Bit	0x6	3
Logical Stop Bit	0x7	0
Physical Start Address	0x8:0x9	0x0F01
Physical Start Bit	0xA	1
Type	0xB	Read and/or Write
Activate	0xC	1 (enabled)



Figuur 3.32: FMMU voorbeeld

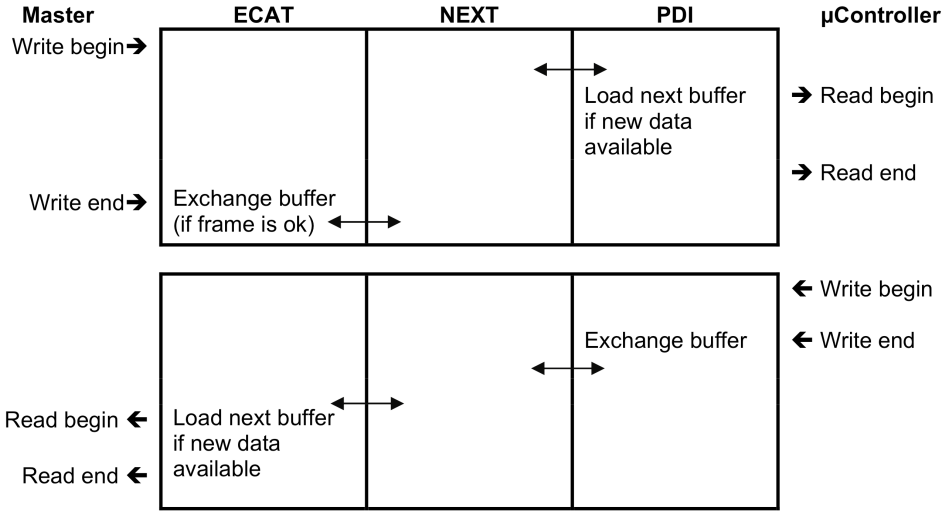
In het voorbeeld geeft de bovenste rij gegevens een deel van de logical process image weer dat verstuurd wordt. De header van het datagram geeft het startadres van dat deel mee. In combinatie met de lengte van het dataveld, dat ook meegegeven wordt in de datagram header, weet de slave precies welk adresbereik in het datagram zit. Uit zijn configuratiegegevens (Tabel 3.4) weet de slave welke bits uit dit deel voor hem bedoeld zijn, en dus welke bits hij moet overnemen in zijn geheugen, dat de onderste rij voorstelt.

3.8.7 SyncManager

De tweezijdige toegang tot het proces- en gebruikersgeheugen kan leiden tot inconsistente data. Dit kan opgelost worden door gebruik te maken van semaforen in de software. Hierdoor moeten zowel de master als de applicatielaag constant het geheugen afpollen om te weten als de gegevens vrij zijn, wat onnodig veel communicatieverkeer vraagt. EtherCAT lost dit op door gebruik te maken van SyncManagers. SyncManagers worden geconfigureerd door de master via de registers van de slave en hebben twee werkingsmodi: Buffered Mode en Mailbox Mode. Een SyncManager wordt geconfigureerd voor een vast adresbereik en datarichting (input of output) in het geheugen van een slave. Afhankelijk van de slave zijn daarom meestal meerdere SyncManagers nodig.

Buffered Mode

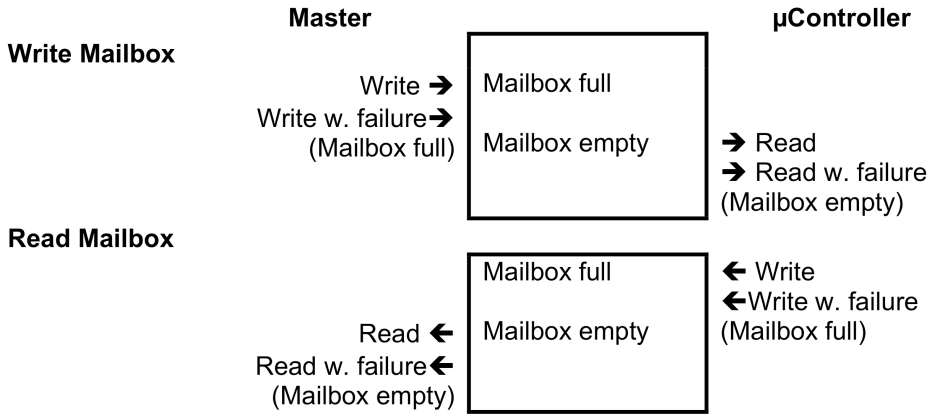
In Buffered Mode is het voor beide zijden op elk moment mogelijk om data te lezen of te schrijven, waardoor deze mode gebruikt wordt voor proces data. Zoals de naam het reeds zegt maakt deze mode gebruik van buffers. Er worden drie evengrote buffers voorzien in het procesgeheugen van de slave, waarvan hun grootte afhankelijk is van de uit te wisselen data. Zowel de master als de applicatie van de slave adresseren steeds het eerste buffer en hebben geen weet van de overige twee buffers. Het is de SyncManager die ervoor zorgt dat de gegevens telkens in een vrij buffer terechtkomen. Als gegevens sneller weggeschreven worden dan uitgelezen, worden de oude gegevens overschreven. Aan de SyncManager wordt ook de datarichting meegegeven (read, write of read/write), zodat de gegevens enkel vanuit de juiste richting kunnen gelezen of geschreven worden. In dat opzicht is er in elke slave een SyncManager voor input data en een SyncManager voor output data. Figuur 3.33 toont de interactie tussen master en slave via buffers.



Figuur 3.33: SyncManager in Buffered Mode

Mailbox Mode

De Mailbox Mode laat enkel alternerend lezen en schrijven van het geheugen toe, waardoor gegevens zeker hun bestemming bereiken. Daarbij kunnen events meegegeven worden die aangeven dat het geheugen geschreven of gelezen is. Dit maakt Mailbox Mode interessant voor parameter data. Om een full-duplex werking mogelijk te maken worden twee Mailbox SyncManagers voorzien, één voor elke richting. Figuur 3.34 toont het principe van SyncManagers in Mailbox Mode.



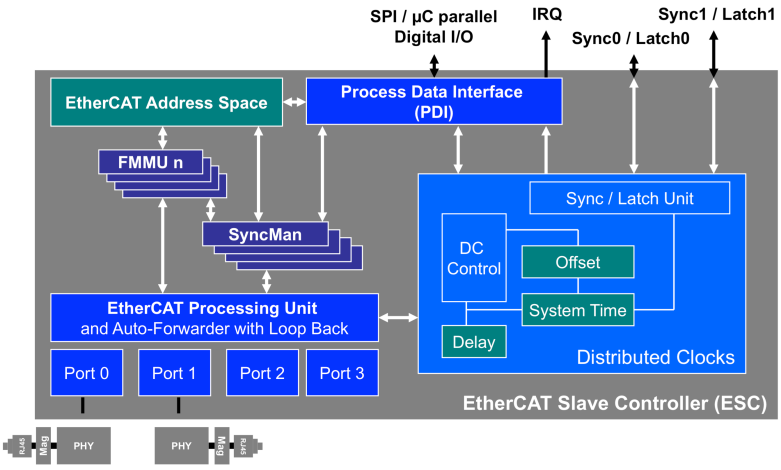
Figuur 3.34: SyncManager in Mailbox Mode

De EtherCAT datagrammen die gegevens voor SyncManagers in Mailbox Mode bevatten worden Mailbox berichten genoemd en worden op applicatieniveau behandeld.

3.8.8 Distributed Clocks

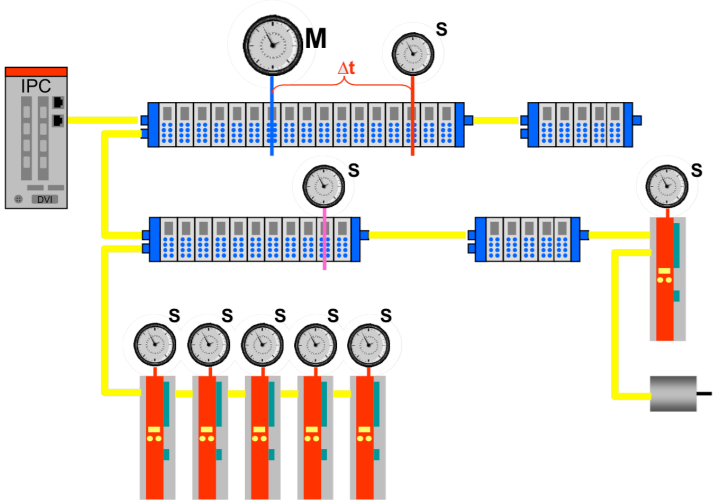
Slaves in een EtherCAT netwerk hebben de mogelijkheid om een lokale klok te implementeren die de huidige tijd opneemt. Als meerdere slaves deze functie bezitten wordt een netwerk van verdeelde klokken gecreëerd. Deze verdeelde klokken worden Distributed Clocks (DC) genoemd en worden gebruikt om de applicatielaag van slave en master te synchroniseren. Eenvoudige slaves zoals digitale en analoge I/O hebben een zeer eenvoudige applicatielaag die veel sneller loopt dan de cyclustijd van EtherCAT en hebben bijgevolg geen synchronisatie nodig. Complexe slaves, zoals bijvoorbeeld drives, hebben meer tijd nodig om gegevens te verwerken waardoor het belangrijk kan zijn om hun cyclustijd af te stemmen op die van de master.

Nauwkeurige synchronisatie van de verschillende klokken is belangrijk om de cyclustijd goed te kunnen afstemmen. Om alle klokken te synchroniseren maakt EtherCAT gebruik van het Precision Time Protocol (PTP), dat beschreven staat in de IEEE 1588 standaard [6]. Door gebruik te maken van dit protocol kunnen de lokale klokken tussen verschillende slaves tot op 100 ns gelijk lopen. Om PTP te implementeren in de slaves die synchronisatie vereisen wordt het Distributed Clocks functieblok voorzien. Figuur 3.35 toont de opbouw van dit functieblok binnen de ESC.



Figuur 3.35: Distributed Clocks binnen de ESC

Om tot de synchronisatie te komen wordt eerst de System Time gedefinieerd. EtherCAT voorziet hiervoor een getal van 64 bits met een basiseenheid van 1 ns. De algemene starttijd (waarde = 0) is gedefinieerd op 1 januari 2000 om 0:00:00. Daarnaast wordt ook een referentieklok aangeduid. Het is gewenst om hiervoor een klok op hardware niveau te gebruiken omdat deze onafhankelijk is van eventuele storingen in software. EtherCAT kiest hiervoor de eerste slave, gezien vanaf de master, die DC ondersteunt. Op die manier kan de System Time met één datagram over het ganse netwerk verdeeld worden. Dit is te zien in figuur 3.36. Als het netwerk ook synchroon met de buitenwereld moet werken, kan gebruik gemaakt worden van een externe IEEE 1588 Grandmaster Clock die de referentieklok instelt.



Figuur 3.36: Principe van Distributed Clocks

Om klokken te synchroniseren worden drie compensaties uitgevoerd:

- **Propagation Delay:** De vertraging van het bericht in de kabel en connector overgangen, gemeten vanaf de master tot de lokale klok van de slave in kwestie.
- **Offset Compensation:** Offset compensation is het vaste tijdsverschil tussen de lokale klok en de referentieklok. Deze afwijking komt door de fout bij de initiële instelling van de klok.
- **Drift Compensation:** Omdat de klokken tellen op basis van een oscillerend kristal zijn er op termijn afwijkingen mogelijk, afhankelijk van de nauwkeurigheid van de oscillator. Om dit te compenseren wordt de System Time cyclisch doorgestuurd. Afhankelijk van het verschil wordt de lokale klok versneld of vertraagd. Dit is een continu proces dat blijft doorlopen teneinde de afwijking van de klokken binnen de perken te houden (typisch 100 ns).

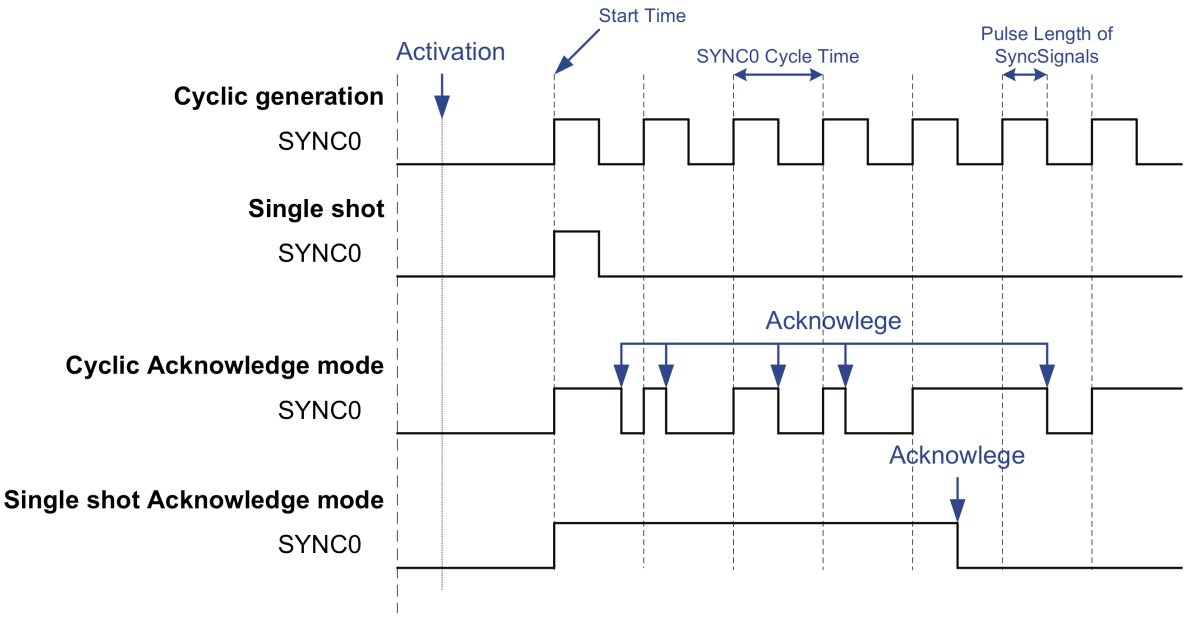
Om de functies van een gesynchroniseerde klok te kunnen gebruiken op applicatieniveau voorziet het DC functieblok twee soorten signalen: Sync signalen en Latch signalen.

Sync signaal

Sync signalen zijn uitgangen van de ESC die hoog worden van zodra de lokale klok een vooraf ingestelde tijd bereikt. Deze uitgangen worden gekoppeld aan de applicatielaag en worden daar gebruikt om uitgangen te activeren of om ingangen weg te schrijven naar het procesgeheugen. Een DC functieblok voorziet twee Sync signalen die afhankelijk zijn van elkaar: Sync0 en Sync1. Sync0 wordt geconfigureerd op basis van een bepaalde starttijd, een cyclustijd en een pulsbreedte. Afhankelijk van deze instellingen wordt een bepaalde werkingsmodus gecreëerd, zoals te zien is in tabel 3.5. Figuur 3.37 toont de betekenis van deze werkingsmodi. Sync1 volgt hetzelfde patroon als Sync0 maar met een bepaalde instelbare vertraging en kan bijvoorbeeld gebruikt worden als follow up.

Tabel 3.5: Configuratie werkingsmodus Sync0

	Sync0 Cycle Time	
Pulse Length	**> 0**	**= 0**
> 0	Cyclic generation	Single shot
= 0	Cyclic Acknowledge	Single shot Acknowledge



Figuur 3.37: Werkingsmodi Sync0

De cyclustijd van het Sync signaal wordt meestal gelijk genomen aan de cyclustijd van het frame dat de procesdata voor de bijhorende applicatie bevat. Op die manier wordt een cyclustijd gecreëerd die gelijkloopt met de master, maar er onafhankelijk van is. Tevens loopt deze cyclus in alle slaves die hetzelfde frame gebruiken gelijk. Hierdoor is het mogelijk om uitgangen in eenzelfde groep (Sync Unit genaamd) binnen een zeer nauwkeurige tijd aan te sturen.

Als extra nauwkeurigheid kan vanuit de master voor elke slave afzonderlijk een Shift Time ingesteld worden. Deze Shift Time houdt rekening met eventuele extra verwerkingstijden van een slave. Zo is er de DCInputShift die ervoor zorgt dat ingangen vóór elke cyclus binnengelezen worden en de DCOutputShift die zorgt dat er gewacht wordt met de uitgangen aan te sturen.

Latch signaal

Een Latch signaal is een ingangssignaal voor het DC functieblok dat gebruikt wordt voor tijdsregistratie. De applicatielaag van de slave stuurt deze signalen aan bij het inlezen van bepaalde ingangen (input Latch), bij het zien van de SFD van een frame of een ander signaal van de applicatielaag. De ESC houdt de tijd waarop het signaal geactiveerd wordt bij. Er zijn verschillende instellingen beschikbaar om deze tijd te loggen: op positieve flank of negatieve flankdetectie, continu of eenmalig.

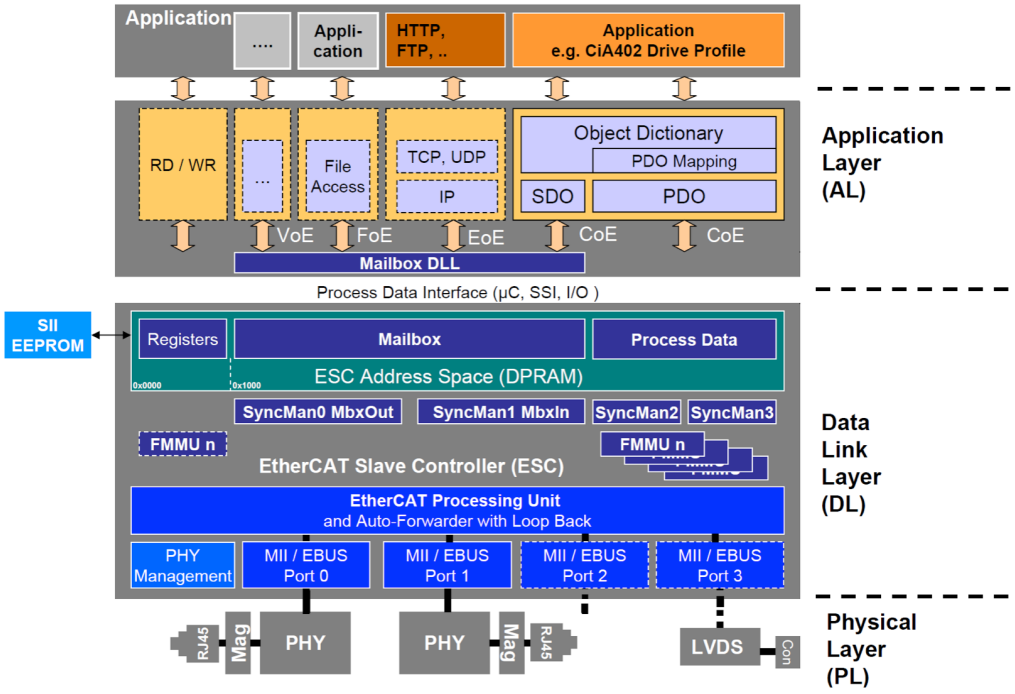
3.8.9 Synchronisatie modi

Afhankelijk van het type slave kan in combinatie met de Distributed Clocks en de SyncManagers een bepaalde vorm van synchronisatie vastgelegd worden. Er worden drie grote categorieën onderscheiden:

- **Free Run:** Slaves zijn niet gesynchroniseerd met de EtherCAT cyclus. Hun input en output update tijd is niet gekend.
- **Synchronous to SM2/3:** De cyclustijd van de slaves wordt geactiveerd op basis van events gegenereerd door SyncManager 2 (Process data output) of SyncManager 3 (Process data input). In deze situatie heeft de master indirect controle over de slave via de cyclustijd van het dataframe. Hierbij moet rekening gehouden worden met vertragingen door de SyncManagers zelf.
- **DC Synchronous:** De slaves worden gesynchroniseerd op basis van Sync0 en Sync1 signalen, welke afhankelijk zijn van de lokale klok van de slave. Deze manier is het meest synchroon met de EtherCAT cyclus en wordt gebruikt in de meest tijdkritische toepassingen.

3.9 Laag 7: Applicatielaag

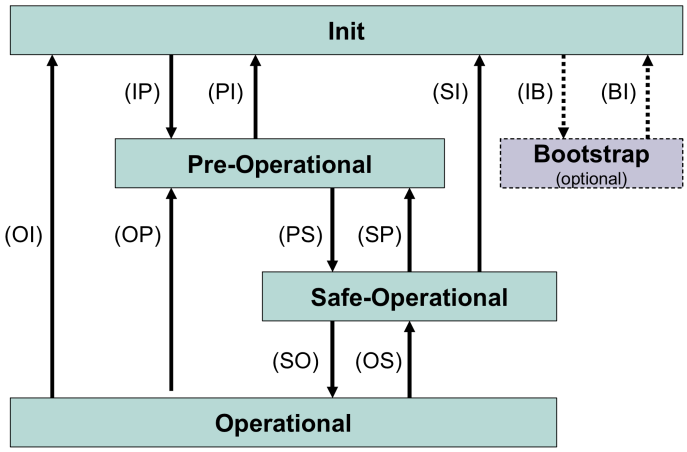
De applicatielaag vormt de schakel tussen de effectieve toepassing van de slave en de ESC op de datalinklaag. Afhankelijk van de toepassing worden verschillende protocollen voorzien die gebruik maken van de Mailbox interface. Figuur 3.38 toont de opbouw van de applicatielaag en zijn plaats binnen het OSI-model.



Figuur 3.38: Applicatielaag EtherCAT slave in het OSI-model

3.9.1 EtherCAT State Machine

De EtherCAT State Machine is verantwoordelijk voor de coördinatie van de communicatie tussen master en slaves. Om eenduidig de toestand van de slave vast te leggen heeft elke slave een state machine. Elke toestand (state) beschrijft welke communicatie mogelijk is en in welke mate de applicatielaag van een slave acties mag uitvoeren. Figuur 3.39 geeft een schematisch overzicht van de verschillende beschikbare toestanden en de mogelijke overgangen ertussen.



Figuur 3.39: EtherCAT State Machine

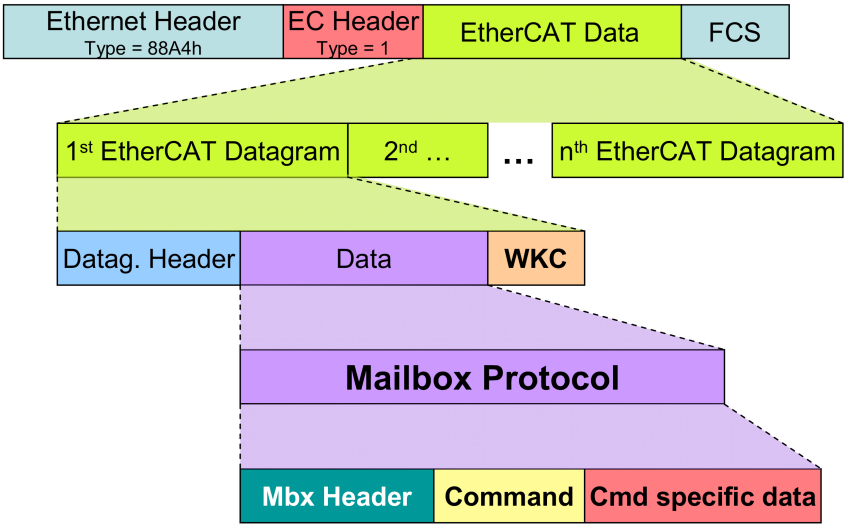
In totaal zijn er vijf verschillende toestanden te onderscheiden:

- **Init:** De applicatielaag heeft geen toegang tot de ESC, de master heeft toegang tot de datalink registers van de slave.
- **Pre-Operational:** Mailbox communicatie is mogelijk, procesdata worden niet verwerkt.
- **Safe-Operational:** Mailbox communicatie is mogelijk en ingangen worden binnen gelezen. Uitgangen worden niet aangestuurd.
- **Operational:** In- en uitgangen zijn geldig en actief, ook Mailbox communicatie is mogelijk.
- **Boot:** Is niet verplicht om te implementeren en wordt gebruikt om firmware updates uit te voeren.

Vanuit de master is het mogelijk om een slave met behulp van states in of uit te schakelen via het Application Layer Control Register.

3.9.2 Mailbox interface

De Mailbox interface wordt gebruikt om data van allerhande protocollen over een EtherCAT netwerk te versturen. Hiervoor wordt in de ESC gebruik gemaakt van SyncManagers volgens het Mailbox principe. Het Mailbox bericht bevindt zich in het dataveld van een EtherCAT datagram, zoals te zien is in figuur 3.40.



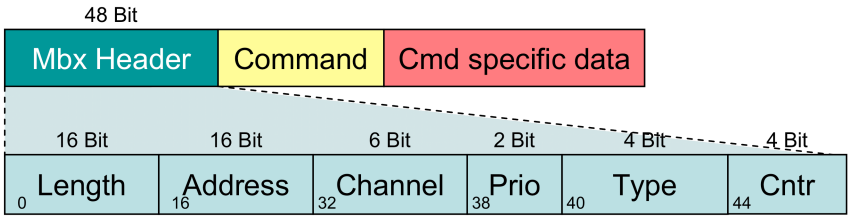
Figuur 3.40: Mailbox binnen het Ethernet frame

Het Mailbox bericht bestaat uit drie velden:

- **Mailbox header (48 bits):** De Mailbox header bevat informatie over het protocol dat in het bericht bevat zit.
- **Command:** Het Command veld bevat de header van het protocol dat in het mailbox bericht zit. De lengte en invulling is afhankelijk van het protocol.
- **Command specific data:** De effectieve data die het gebruikte protocol wil versturen. De maximale lengte is afhankelijk van de lengte van het Command veld en dus afhankelijk van het protocol.

Mailbox header

Op basis van de Mailbox header wordt de inhoud van een Mailbox bericht doorgestuurd naar het corresponderende protocol op applicatieniveau. Figuur 3.41 toont de opbouw van de Mailbox header.



Figuur 3.41: Mailbox header

De Mailbox header bestaat uit zes velden:

- **Length (16 bits):** De lengte van de data binnen het Mailbox bericht in bytes.
- **Address (16 bits):** Het vaste Node Address van de verzender van het bericht.
- **Channel (6 bits):** Wordt nog niet gebruikt, is voorlopig 0.
- **Priority (2 bits):** Het prioriteitsniveau tussen 0 (laagste) en 3 (hoogste).
- **Type (4 bits):** Het gebruikte Mailbox protocol. Tabel 3.6 geeft een overzicht van de gebruikte protocollen.

- **Counter (4 bits):** Het sequentie nummer, wordt gebruikt bij meerdere opeenvolgende frames.

Tabel 3.6: Mailbox types

Waarde (DEC)	Mailbox protocol
0	Error
1	ADS over EtherCAT (AoE)
2	Ethernet over EtherCAT (EoE)
3	CANopen over EtherCAT (CoE)
4	File access over EtherCAT (FoE)
5	Servo profile over EtherCAT (SoE)
15	Vendor specific (VoE)

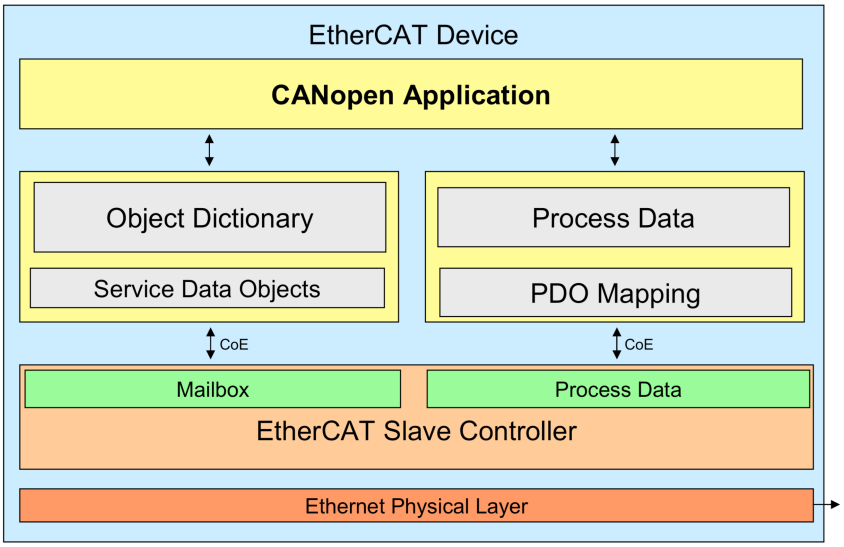
3.9.3 Ethernet over EtherCAT

Ethernet over EtherCAT (EoE) laat toe om standaard Ethernet berichten te verzenden naar EtherCAT slaves. Dit kunnen standaard Ethernet TCP/IP berichten zijn. Hierdoor kunnen bijvoorbeeld over het internet de gegevens van een intelligente slave rechtstreeks uitgelezen worden, zonder dat de master deze apart moet loggen. Een EtherCAT segment kan aan een lokaal netwerk gekoppeld worden via een speciale EtherCAT switch of een switchport terminal. Deze toestellen zullen berichten die bedoeld zijn voor een EtherCAT toestel in een EoE Mailbox bericht plaatsen en doorgeven in het one-total-frame. De master zal dit bericht ontvangen en aan de hand van het destination MAC-adres wordt het bericht doorgestuurd naar een specifieke slave via het Node Address. Het is de taak van de master om als virtuele switch te werken en de MAC-adressen van de slaves met EoE functionaliteit bij te houden. De speciale switches en switchport terminals hebben enkel de functie om een standaard Ethernet TCP/IP bericht te verpakken in een EtherCAT Mailbox bericht.

3.9.4 CANopen over EtherCAT

Complexere slaves hebben naast procesdata ook parameter data die niet cyclisch verstuurd moeten worden. Voor zo'n slaves is de CANopen object architectuur interessant om te gebruiken. Daarom voorziet EtherCAT een eenvoudige integratie van de CANopen protocolstack via het CANopen over EtherCAT (CoE) protocol.

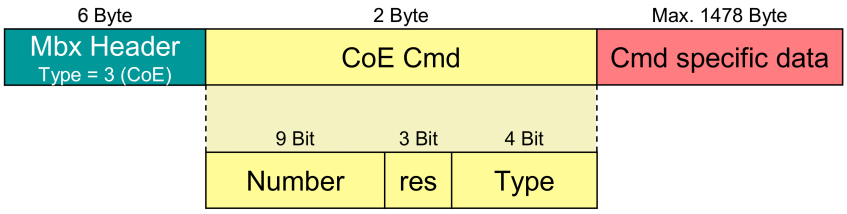
CANopen maakt een onderscheid tussen Service Data Objects (SDO's) voor parameter data en Process Data Objects (PDO's) voor procesdata. In EtherCAT worden de SDO's geadresseerd via het Mailbox protocol terwijl PDO's rechtstreeks gelinkt worden aan de procesdata in het DPRAM, zoals te zien is in figuur 3.42.



Figuur 3.42: Protocolstack CoE

CoE header

Het command veld van CoE in het Mailbox bericht is weergegeven in figuur 3.43.



Figuur 3.43: CoE header

De header bestaat uit drie velden:

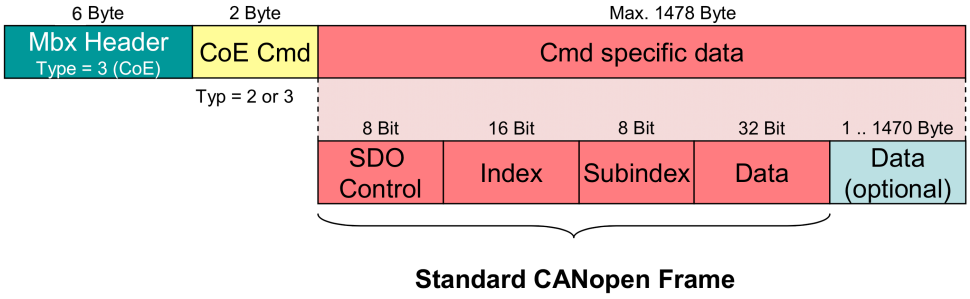
- **Number (9 bits):** Het PDO nummer, indien PDO gegevens geadresseerd worden.
- **Reserved (3 bits):** Gereserveerde bits.
- **Type (4 bits):** Het soort CANopen bericht. Tabel 3.7 geeft een overzicht van de verschillende types CoE berichten. Aangezien procesdata rechtstreeks doorgestuurd worden, zijn meest gebruikte types SDO Request, SDO Response en SDO Information.

Tabel 3.7: CoE type veld

Waarde (DEC)	CoE berichttype
0	Gereserveerd
1	Emergency message
2	SDO Request
3	SDO Response
4	TxPDO
5	RxPDO
6	Remote transmission request of TxPDO
7	Remote transmission request of RxPDO
8	SDO Information
9-15	Nog niet in gebruik

CoE dataveld

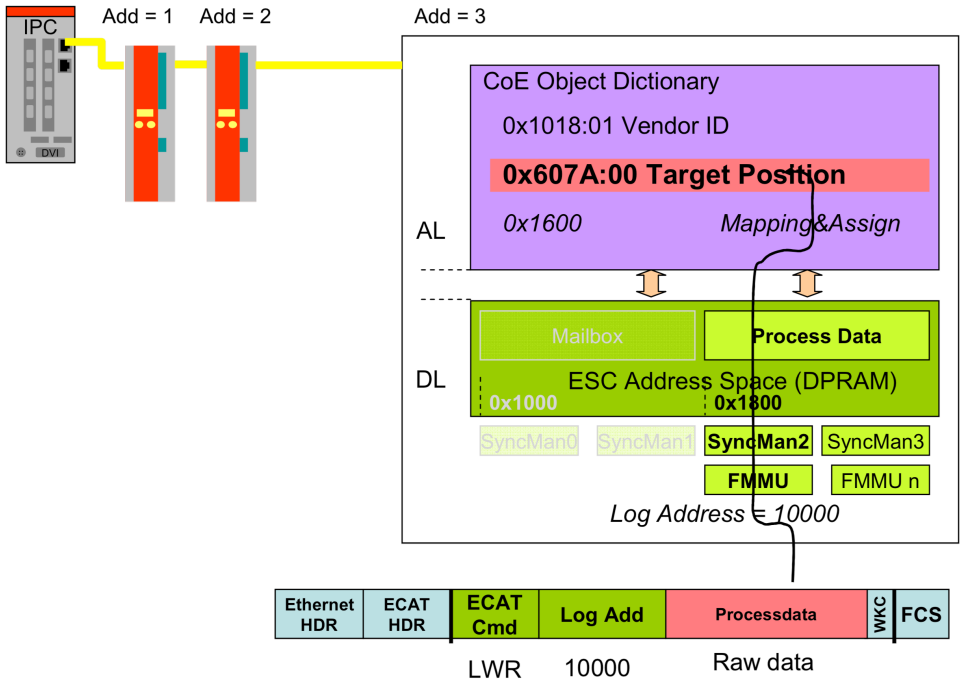
Het CoE dataveld bestaat uit een standaard CANopen frame aangevuld met een extra dataveld, zoals te zien is in figuur 3.44. Dit dataveld zorgt ervoor dat er meer gegevens dan de standaard 32 databits van CANopen verstuurd kunnen worden.



Figuur 3.44: CoE dataveld

Voorbeeld van PDO datatoegang

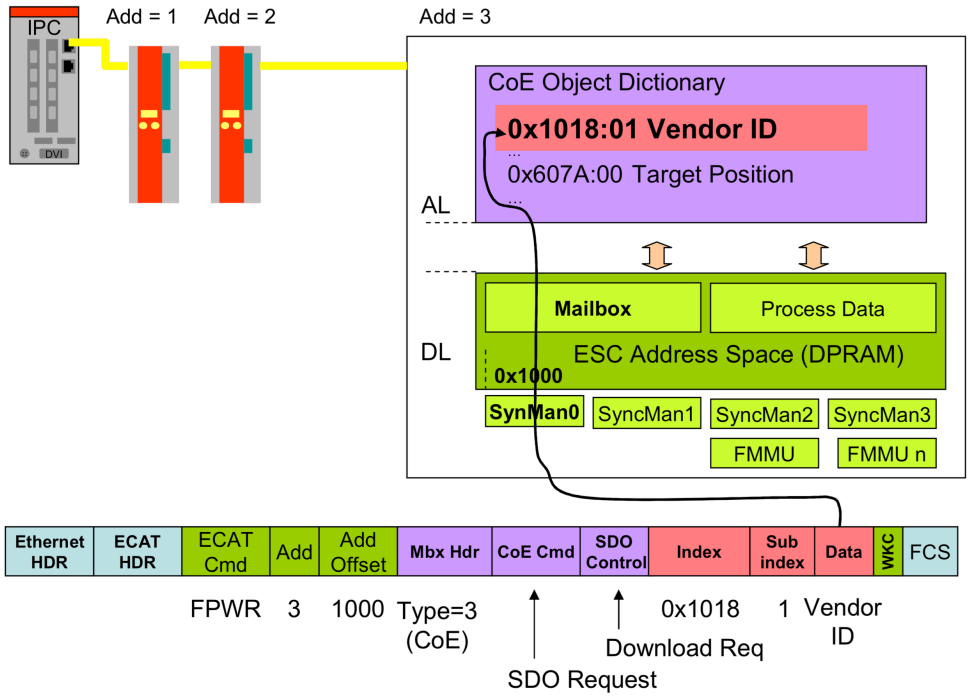
Figuur 3.45 toont hoe procesdata via CoE verwerkt worden. Door het rechtstreeks linken van de PDO's aan de procesdata in het DPRAM kan gebruik gemaakt worden van FMMU's en dus van Logical Addressing. Op die manier kunnen CoE slaves realtime procesdata ontvangen en verwerken.



Figuur 3.45: PDO mapping CoE

Voorbeeld van SDO datatoegang

SDO's bevatten parameter data die niet cyclisch verstuurd moeten worden. Deze data bevinden zich bijgevolg niet in de logical process image en kunnen dus niet via logische adressering verstuurd worden. Er moet dus gebruik gemaakt worden van het Mailbox protocol, zoals te zien is in figuur 3.46. SyncManagers in Mailbox Mode zorgen ervoor dat de gegevens consistent zijn.



Figuur 3.46: SDO mapping CoE

4 Praktische demo applicatie

4.1 Inleiding

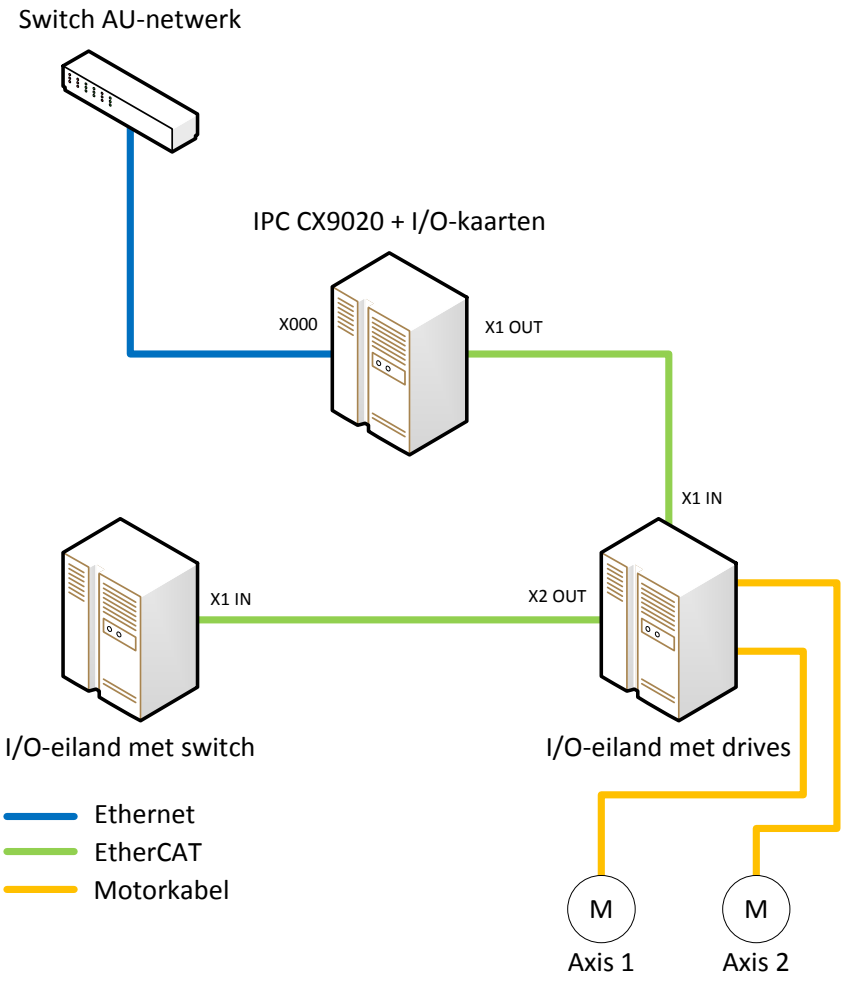
Om de technische kant van EtherCAT aan te tonen wordt een demo applicatie gebouwd. Aan de hand van een stappenplan kan de uitvoerder kennismaken met functies zoals redundantie, HotConnect en CoE. Aangezien EtherCAT met zijn Distributed Clocks functie focust op de nauwkeurige synchronisatie van het netwerk, is het uiteindelijke doel van de opstelling de synchronisatie van twee motoren.

4.2 Proefopstelling

Om de opstelling te bouwen werd door XiaK het volgende materiaal van Beckhoff ter beschikking gesteld:

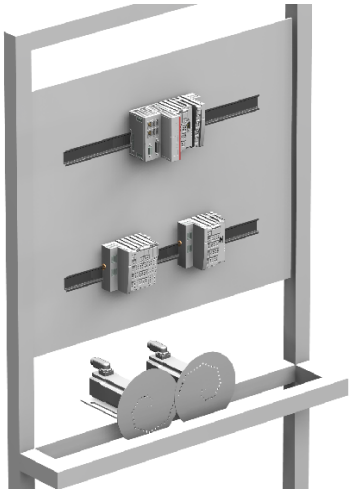
- IPC CX9020
- Servo Motor Terminal EL7201 (2x)
- DC Servo Motor AM8121-0F21 (2x)
- Buffer Capacitor Terminal EL9570 (2x)
- Analog Input EL3062 (2x)
- Analog Output EL4002
- 8 Digital In/8 Digital Out EL1859 (2x)
- 4 Port Ethernet Switch Terminal EL6614
- CANopen master EL6751
- EtherCAT Coupler EK1100 (2x)
- EtherCAT Extension Terminal EK1110
- Bus End Plate EL9011 (2x)

Na het bestuderen en testen van de hardware werd gekozen voor een configuratie die het mogelijk maakt om verschillende eigenschappen van EtherCAT aan te tonen. Een schematisch overzicht van de uiteindelijke opstelling is weergegeven in figuur 4.1.



Figuur 4.1: Schematische voorstelling proefopstelling

Om de opstelling mooi te presenteren wordt de hardware bevestigd op een mobiele demostand. Hiervoor werden enkele aangepaste onderdelen ontworpen die het mogelijk maken de motoren vrij te positioneren op de opstelling. Uiteindelijk werd met behulp van het softwarepakket Siemens NX een 3D-model van de opstelling gemaakt, zoals weergegeven in figuur 4.2.



Figuur 4.2: 3D-model demo-opstelling

4.3 Inhoud van de proef

4.3.1 Inleiding TwinCAT 3

De eerste stap in de proef is een kennismaking met de programmeeromgeving TwinCAT (Figuur 4.3). TwinCAT staat voor The Windows Control Automation Technology en geeft realtime functionaliteit aan een Windows-systeem. Hierbij wordt een onderscheid gemaakt tussen TwinCAT eXtended Automation Engineering (XAE) en de TwinCAT runtime.

Figuur 4.3: TwinCAT 3

TwinCAT XAE is de programmeeromgeving om een EtherCAT systeem te configureren en te programmeren en is gebaseerd op Microsoft Visual Studio. De TwinCAT runtime daarentegen wordt gebruikt om het besturingssysteem van de IPC (Windows) cyclisch te onderbreken om de realtime functionaliteiten, zoals de PLC task, uit te voeren.

4.3.2 Redundantie

Nadat de uitvoerder kennis heeft gemaakt met de programmeeromgeving wordt netwerkredundantie toegelicht. Zoals vermeld in het theoretisch deel voorziet EtherCAT redundantie via een tweede Ethernet-poort op de master. Het is de bedoeling om een redundante ringstructuur op te bouwen.

Via de programmeeromgeving kan de netwerktopologie visueel weergegeven worden. Door het monitoren van de encoder variabele van één van de motoren kan de redundantie aangetoond worden. Bij het verwijderen van een willekeurige kabel wordt deze variabele nog steeds realtime aangepast. Wordt een tweede kabel verwijderd, dan valt de communicatie weg.

4.3.3 Analyseren van berichten

Aangezien EtherCAT datagrammen verpakt worden in een standaard Ethernet frame kunnen ze gemonitord worden met een lokale PC. Hiervoor is het gratis softwarepakket Wireshark nodig.

Doordat het EtherCAT segment aangesloten is op de E-Bus van de master, wordt in Direct Mode gewerkt. Dit wil zeggen dat de master geen MAC-adres nodig heeft om zijn EtherCAT datagrammen te versturen. Het MAC-adres van de bestemmeling wordt een multicast. Hierdoor kunnen de EtherCAT datagrammen ook verspreid worden over een standaard Ethernet netwerk en kunnen ze dus door de lokale PC ontvangen worden. In Open Mode gebruikt de master het specifieke MAC-adres van de buskoppelaar, waardoor andere toestellen op het netwerk de frames niet ontvangen. In zo'n situatie is het mogelijk om de master een tweede frame te laten versturen met een multicast adres.

Door gebruik te maken van de redundante structuur wordt één van de RJ-45 connectoren gebruikt door het EtherCAT segment. De IPC heeft twee zo'n poorten maar slechts één MAC-adres voor beide. Een tweede MAC-adres wordt gebruikt door het E-Bus segment. Doordat één van de RJ-45 connectoren deel uitmaakt van het EtherCAT segment, worden alle EtherCAT berichten (zowel doorgaand als terugkerend) hierover verstuurd en kunnen ze dus door de lokale PC ontvangen worden. Hoe de IPC dit regelt is ongekend. Het is mogelijk dat de IPC bij een redundante structuur zijn beide MAC-sublagen aan elkaar koppelt. Dit zou inhouden dat alle standaard Ethernet berichten voor de IPC ook over het EtherCAT segment verstuurd worden. Op zich zou dit geen probleem zijn,

aangezien EtherCAT slaves standaard alle andere berichten dan EtherCAT tegenhouden.

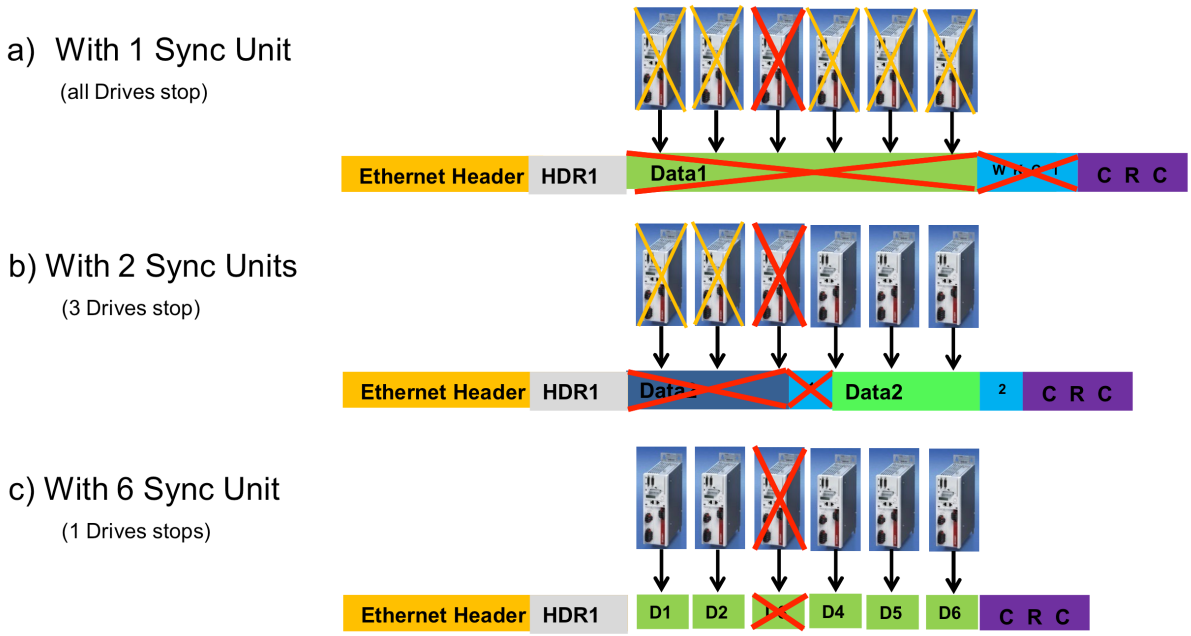
Een onderscheid tussen de doorgaande en terugkerende frames kan gemaakt worden aan de hand van de Auto-Increment Addressing berichten. Het doorgaande bericht heeft hierbij een negatieve waarde, het terugkerende een positieve.

4.3.4 Sync Units

Sync Units zijn gegroepeerde slaves in een EtherCAT netwerk die hetzelfde Ethernet frame gebruiken en, indien mogelijk, hetzelfde EtherCAT datagram. Ze worden met hetzelfde cyclisch interval aangesproken en zijn dus "gesynchroniseerd", een basisvereiste voor de Distributed Clocks functionaliteit. Standaard worden alle slaves in de Default Sync Unit geplaatst.

Een andere reden om bepaalde slaves al dan niet in dezelfde Sync Unit te plaatsen is redundantie. Slaves in dezelfde Sync Unit gebruiken (voor bepaalde data) hetzelfde datagram en hebben bijgevolg dezelfde Working Counter (WKC). Als bepaalde gegevens om één of andere reden niet weggeschreven of ingelezen worden, dan klopt de WKC van het datagram niet meer en detecteert de master een fout. De master zal proberen het datagram opnieuw te versturen aan de hand van zijn index. Als dit niet lukt wordt de slave op inactief geplaatst. De master heeft met de WKC geen informatie over de specifieke slave die in fout gaat en bijgevolg worden alle slaves die met dat datagram worden aangesproken uitgeschakeld.

Stel dat bepaalde slaves altijd samen moeten uitschakelen, dan kunnen deze in dezelfde Sync Unit geplaatst worden. Omgekeerd, mogen bepaalde slaves niet zomaar stilvallen, dan kunnen deze in een aparte Sync Unit geplaatst worden. Figuur 4.4 toont dit principe. Een WKC-fout kan niet gesimuleerd worden waardoor het praktisch testen van dit principe niet mogelijk is.



Figuur 4.4: WKC Diagnose

4.3.5 HotConnect

HotConnect is een praktisch aspect van EtherCAT. Verschillende I/O-eilanden kunnen als HotConnect gedefinieerd worden, wat inhoudt dat deze op eender welk moment en op eender welke plaats in het netwerk verwijderd

en aangesloten mogen worden. Als de modules Fast HotConnect ondersteunen kunnen zelfs verschillende I/O-eilanden met hetzelfde type hardware onderling uitgewisseld worden.

Bij standaard HotConnect, zoals in de proefopstelling, wordt aan elk I/O-eiland een identificatienummer toegekend. Dit wordt opgeslagen in het EEPROM geheugen van de buskoppelaar. Bij Fast HotConnect wordt het identificatienummer via fysieke switches op de buskoppelaar ingesteld. In de proefopstelling worden beide I/O-eilanden als HotConnect gedefinieerd. Net als bij redundantie kan de werking aangetoond worden door het monitoren van enkele variabelen.

4.3.6 CoE

Integratie van CANopen is een belangrijk aspect van EtherCAT en wordt CANopen over EtherCAT (CoE) genoemd. Er wordt een onderscheid gemaakt tussen procesdata (PDO's in CANopen) en parameter data (SDO's in CANopen). EtherCAT voorziet de mogelijkheid om PDO's te versturen via Logical Addressing. SDO's worden acyclisch verstuurd (op verzoek van de master) en worden verpakt in een Mailbox bericht. In de proefopstelling werken de drives volgens CoE. De PDO's en SDO's van de drives worden bekeken, zodat het CANopen profiel van deze toestellen duidelijk wordt. Het is eventueel ook mogelijk om Mailbox berichten met SDO informatie te analyseren met behulp van Wireshark.

4.3.7 Numerical Control

Om de motoren met bijhorende drives eenvoudiger in gebruik te nemen voorziet TwinCAT Numerical Control (NC). Dit onderdeel, dat deel uitmaakt van het Motion geheel, voorziet een eigen task en wordt gekoppeld aan een drive. De interface van een drive is eerder complex: via het Controlewoord kan de drive in verschillende werkingsmodi geplaatst worden en het Statuswoord geeft feedback van de drive weer. NC ontleedt deze informatie en geeft ze duidelijk weer.

Aan het onderdeel NC kunnen verschillende motoren toegevoegd worden. Deze worden aangeduid met de naam Axis. Een Axis wordt opgedeeld in drie delen: Encoder, Drive en Controller. In het onderdeel Encoder kunnen alle parameters die verband houden met de encoder ingesteld worden, zoals bijvoorbeeld het nulpunt instellen. Het onderdeel Drive bevat parameters over de drive zelf, zoals bijvoorbeeld maximale output en filterinstellingen. Bij het onderdeel Controller kunnen alle regeltechnische parameters aangepast worden, zoals bijvoorbeeld de K-factor van de positieregelaar.

De NC zelf heeft een interface die gekoppeld kan worden aan een PLC programma. Om die koppeling eenvoudiger te maken is er de bibliotheek TC2_MC2. In plaats van elke variabele van de interface afzonderlijk aan te spreken, wordt in met deze bibliotheek alle informatie gebundeld in één input/output variabele. Via speciale functieblokken uit de bibliotheek kunnen bepaalde bewegingen aangelegd worden. Dit zal de basis vormen voor de synchronisatie van de motoren.

4.3.8 PLC

Om PLC-software te programmeren voorziet TwinCAT het onderdeel PLC. In het stappenplan wordt toegelicht hoe de klassieke functies en functieblokken aangemaakt en geprogrammeerd kunnen worden, hoe variabelen aangemaakt worden en hoe bibliotheken toegevoegd kunnen worden.

Daarnaast bevat dit onderdeel ook de mogelijkheid om visualisaties aan te maken. Deze visualisaties kunnen op bepaalde IPC's rechtstreeks weergegeven worden. Echter, de IPC in de opstelling is een Compact Edition (CE) die dit niet ondersteunt. Visualisaties kunnen wel gesimuleerd worden op de lokale PC.

4.3.9 Synchronisatie van motoren

De uiteindelijke uitdaging van de proefopstelling is de synchronisatie van de motoren. Omdat de opstelling robuust moet zijn, aangezien deze door studenten uitgevoerd moet worden, kan de nauwkeurige aansturing niet zomaar aangetoond worden. Het is met het oog niet mogelijk om vast te stellen dat beide motoren tot op minder dan 1 μs gelijktijdig aangestuurd worden. De enige manier om de nauwkeurigheid visueel aan te tonen is met een opstelling die erg kritische bewegingen moet uitvoeren. Een voorbeeld is tandwielen in elkaar laten draaien, waarbij een fout mogelijk leidt tot een breuk. Dit kan uiteraard niet gebruikt worden in deze opstelling. Een andere mogelijkheid is het gebruik van heel veel motoren. Dankzij EtherCAT kunnen grote aantallen zeer snel en nauwkeurig aangestuurd worden. In de opstelling zijn slechts twee motoren beschikbaar waardoor dit niet mogelijk is. Er moet dus een alternatief gezocht worden.

Dat alternatief werd gevonden door gebruik te maken van een laser. Op beide motorassen is een schijf met een bepaald gatenpatroon bevestigd. Op deze schijven wordt een lijnlaser gericht die door de gaten schijnt. Als de motoren synchroon draaien wordt een lichtpatroon weergegeven die de toeschouwer duidelijk maakt dat de motoren inderdaad synchroon draaien (gelijke hoek en gelijke snelheid). Deze methode heeft zijn beperkingen: de nauwkeurigheid is afhankelijk van de fysieke opstelling van de motoren en de laser. Daarbij komt dat het lichtpatroon enkel zichtbaar is als de motoren op lage snelheid draaien.

Nu de synchronisatie van motoren visueel weergegeven kan worden, werd gezocht hoe de synchronisatie moet verlopen. Een eerste mogelijkheid is het gelijktijdig starten van de motoren, wat mogelijk gemaakt wordt door de Distributed Clocks functie. Deze nauwkeurigheid kan echter met het blote oog niet vastgesteld worden.

Om de studenten die de opstelling moeten programmeren een uitdaging te geven en om een visueel aantrekkelijker effect te creëren, werd gekozen om de motoren al draaiend te laten synchroniseren. Hierbij worden de motoren elk op een willekeurig moment gestart. Van zodra een knop bediend wordt, worden de motoren gesynchroniseerd. Hierbij wordt de motor die achterloopt versneld tot deze op dezelfde positie zit als de eerste motor. Via de laser kan vastgesteld worden dat de motoren initieel niet synchroon draaien en na de synchronisatie wel. Als extra worden de motoren bij stilstand altijd volgens dezelfde hoek geplaatst. Deze hoek is zodanig gekozen dat de toeschouwer duidelijk kan vaststellen dat de motoren onder dezelfde hoek staan.

5 Besluit

EtherCAT is een performante veldbus die gebaseerd is op Ethernet. Hiervoor berust het op twee basisprincipes: master-slave en one-total-frame. Het master-slave principe lost het probleem van de busarbitrage op, terwijl het one-total-frame principe zorgt voor hoge efficiëntie.

Om de speciale functies van EtherCAT, zoals het on-the-fly verwerken van gegevens, te kunnen implementeren, voorziet EtherCAT de EtherCAT Slave Controller (ESC). Deze chip wordt om de functies van de datalinklaag in te vullen. Hierdoor kan een EtherCAT netwerk niet rechtstreeks aangesloten worden op een Ethernet netwerk. Er is een speciale EtherCAT switch nodig, wat een extra kost met zich meebrengt.

Als de koppeling naar een standaard Ethernet netwerk toch gelegd moet worden, kan het EtherCAT Automation Protocol (EAP) gebruikt worden. Dit protocol, dat niet verward mag worden met EtherCAT, wordt volledig geïntegreerd in een standaard Ethernet TCP/IP frame. De toepassingen zijn onder andere master-master communicatie en het communiceren met SCADA en HMI toepassingen, waarbij realtime werking niet de essentie is. Het voordeel van het EAP protocol is de mogelijkheid om over een EtherCAT netwerk verstuurd te worden. Op die manier kunnen toestellen over verschillende types netwerken heen met elkaar communiceren. Aangezien EAP een volledig ander protocol is, werd het niet bestudeerd in deze masterproef en wordt bijgevolg enkel het bestaan ervan vermeld.

Uit de theoretische studie blijkt dat EtherCAT zichzelf profileert als een uitstekende veldbus voor synchronisatie. Hiervoor maakt het gebruik van het Precision Time Protocol (PTP), waardoor lokale klokken in het netwerk tot op 100 ns nauwkeurig gesynchroniseerd worden. Om deze lokale klokken nuttig te gebruiken voorziet EtherCAT het Distributed Clocks (DC) functieblok in de ESC. Afhankelijk van de toepassing kunnen verschillende werkingsmodi ingesteld worden.

Om de praktische kant van EtherCAT aan te tonen werd een demo applicatie gebouwd. Met het beschikbaar gestelde materiaal kunnen enkele speciale eigenschappen van de veldbus gedemonstreerd worden. De grootste uitdaging bevindt zich echter in het aantonen van de synchronisatie. Aangezien studenten de opstelling zelf moeten kunnen programmeren, moet deze robuust zijn. Om dit op te lossen wordt gewerkt met een laser en een gatenpatroon op de motorassen, met als resultaat een lichteffect als de motoren synchroon draaien.

Het probleem met deze aanpak is het aantonen van de nauwkeurigheid. EtherCAT maakt het mogelijk om de motoren gelijktijdig aan te sturen met een zeer hoge nauwkeurigheid. Het is echter niet mogelijk om, met de huidige opstelling, visueel vast te stellen hoe nauwkeurig dit werkelijk is. Het verschil tussen synchroon en asynchroon moet voldoende groot zijn om duidelijk zichtbaar te zijn. Daarom worden de motoren asynchroon gestart om daarna gesynchroniseerd te worden. Op die manier is er niet alleen een uitdaging om de opstelling te programmeren maar is de opstelling ook visueel aantrekkelijker voor toeschouwers.

Achteraf blijkt echter dat EtherCAT weinig te maken heeft met de nauwkeurigheid van de opstelling. Ten eerste hebben de motoren een incrementele encoder met een resolutie van 20 bits. In de motoren is een geheugen voorzien die de positie van de motoren bijhoudt. TwinCAT voorziet de regelkring die het mogelijk maakt om die positie te regelen. De nauwkeurigheid van de opstelling is uiteindelijk afhankelijk van de nauwkeurigheid van de feedback (de encoder) en afstelling van de regelkring. EtherCAT is echter wel in staat om grote hoeveelheden motoren op deze manier aan te sturen. Daarbij is het mogelijk om deze allemaal gelijktijdig te starten, bij te regelen en te stoppen, iets wat met een standaard veldbus niet mogelijk is.

Na het bouwen van de opstelling werd een potentieel betere oplossing bedacht, waarbij EtherCAT wel de oorzaak van de synchronisatie is en waarbij dat visueel weergegeven kan worden. Door gebruik te maken van een scoop met 2 kanalen kunnen de motorsignalen tussen drive en motor opgemeten worden, waardoor het tijdsverschil tussen de startpulsen visueel weergegeven kan worden. Nadeel van deze aanpak is dat ze eenvoudig te programmeren is en dus weinig uitdaging biedt voor studenten.

Literatuurlijst

[1] S. Lammermann, ``Ethernet as a real-time technology.'' University of Telecommunications, Leipzig, June 2008.

[2] E. T. Group, ``http://www.ethercat.org/.'' Online.

[3] Beckhoff GmbH, Eiserstr. 5, *Hardware Datasheet ET1100*, 1.8 ed., Mei 2010.

[4] IEEE, ``Ieee standard for ethernet.'' White Paper, December 2012.

[5] IEEE, ``Ieee standard for information technology: Part 2: Logical link control.'' White Paper, 1998.

[6] IEEE, ``Ieee1588.'' Online, 2010.