EtherCAT Communication

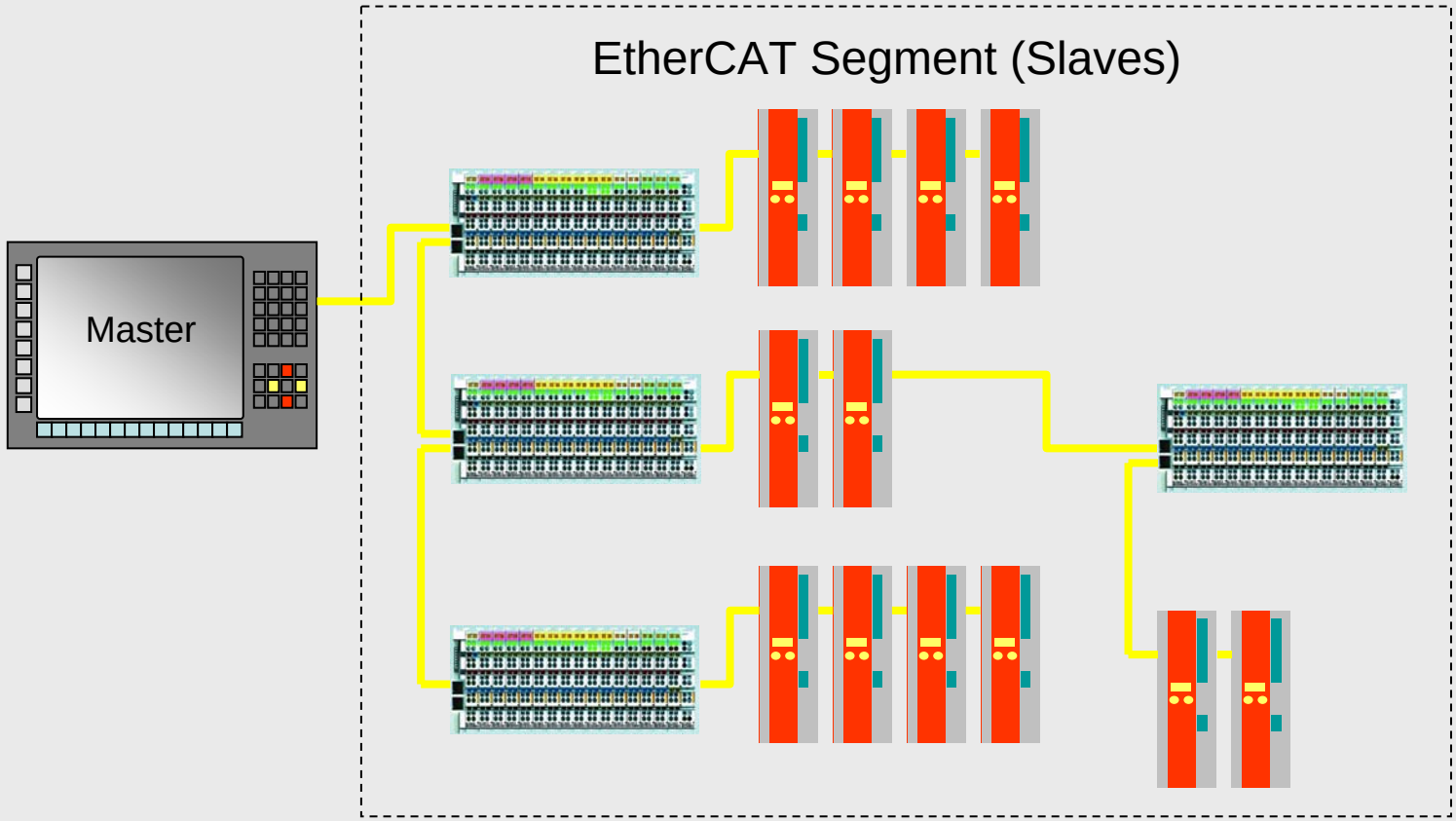
Communication Principles

Agenda

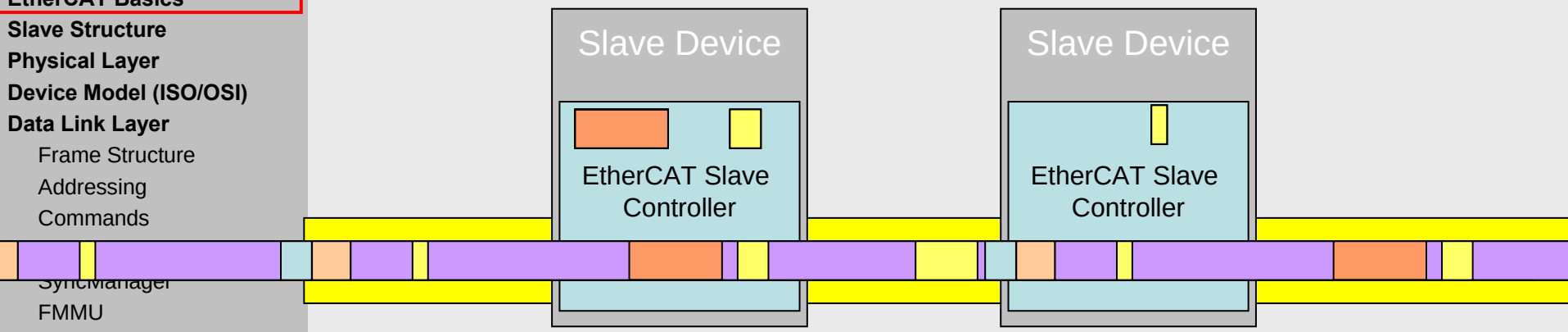
- EtherCAT Basics
- Slave Structure
- Physical Layer
- Device Model
- Data Link Layer
  - Frame Structure
  - Addressing, Commands
  - Memory, SyncManager, FMMUs
  - Diagnosis
- Application Layer
  - State Machine
  - Mailbox (Mailbox Protocols)
  - Slave Information Interface (EEPROM)
- Device Profiles
- Distributed Clocks
- Device Description
- Tools (Configuration Tool, Monitor, …)
- EtherCAT Master
- Standard & References

Topology

- Flexible Topology
- Any number of physical layer changes possible
- Standard Ethernet 100m cable distance between 2 devices
- Up to 65.535 devices possible

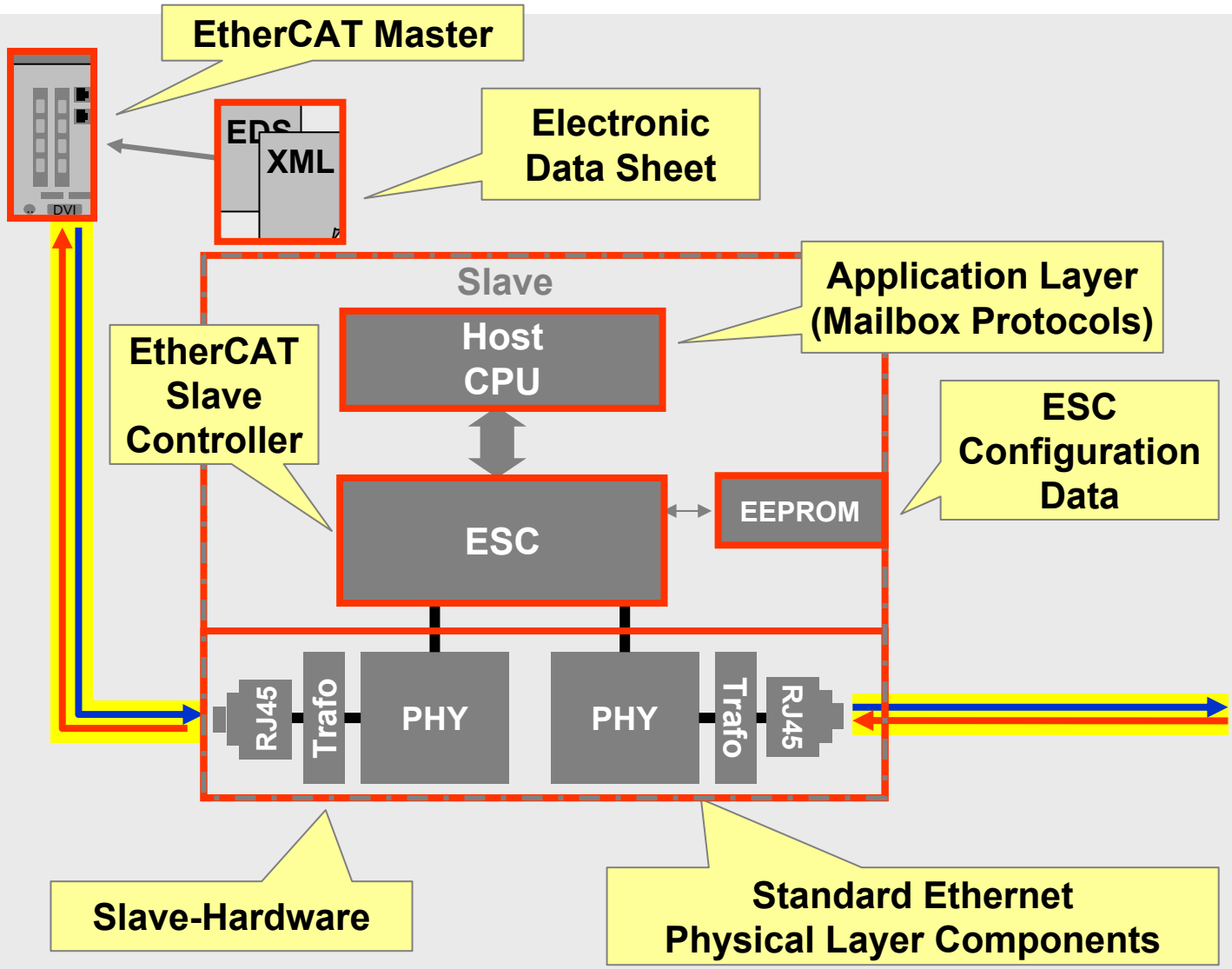


Functional Principle: Ethernet „on the Fly“

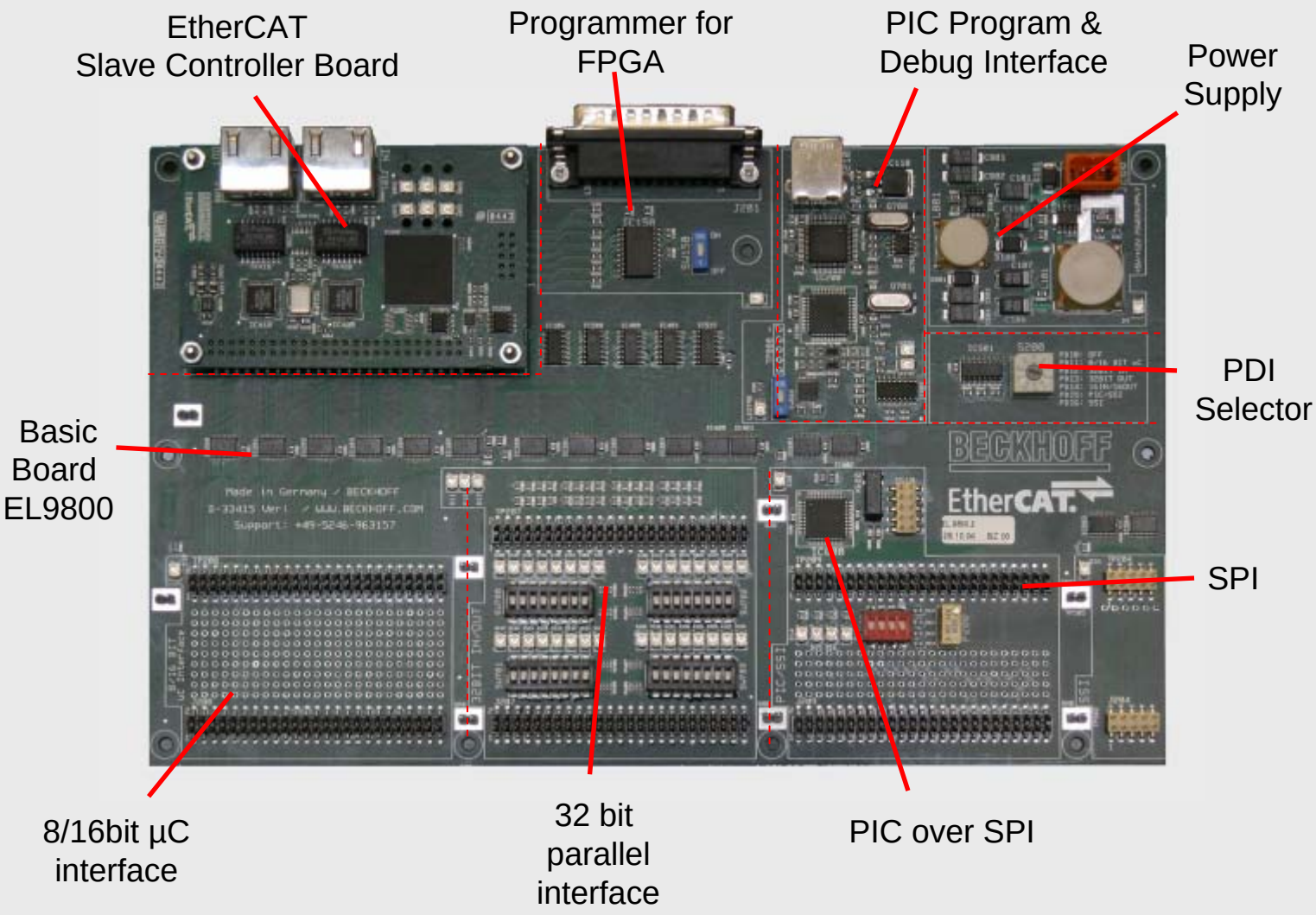


- Process data is extracted and inserted on the fly
- Process data size per slave almost unlimited (1 Bit…60 Kbyte, if needed using several frames)
- Compilation of process data can change in each cycle, e.g. ultra short cycle time for axis, and longer cycles for I/O update possible
- In addition asynchronous, event triggered communication

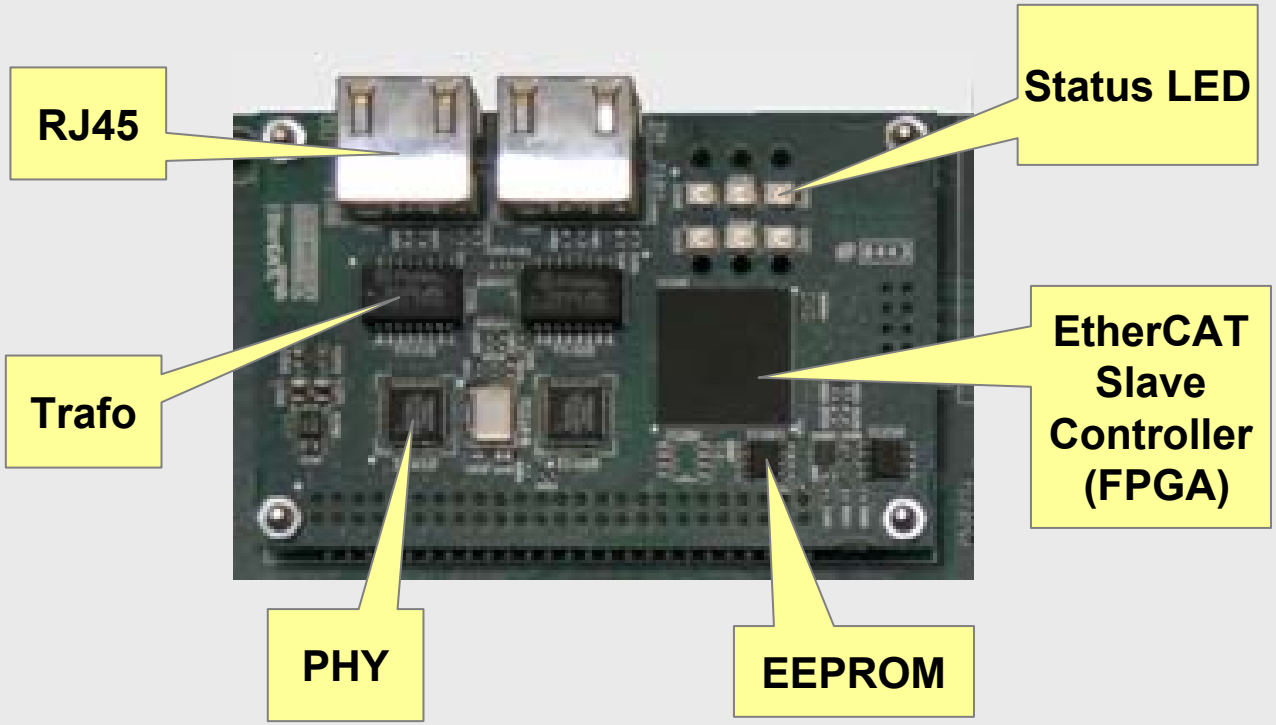
EtherCAT Slave Structure



EtherCAT Slave Evaluation Kit (by Beckhoff)



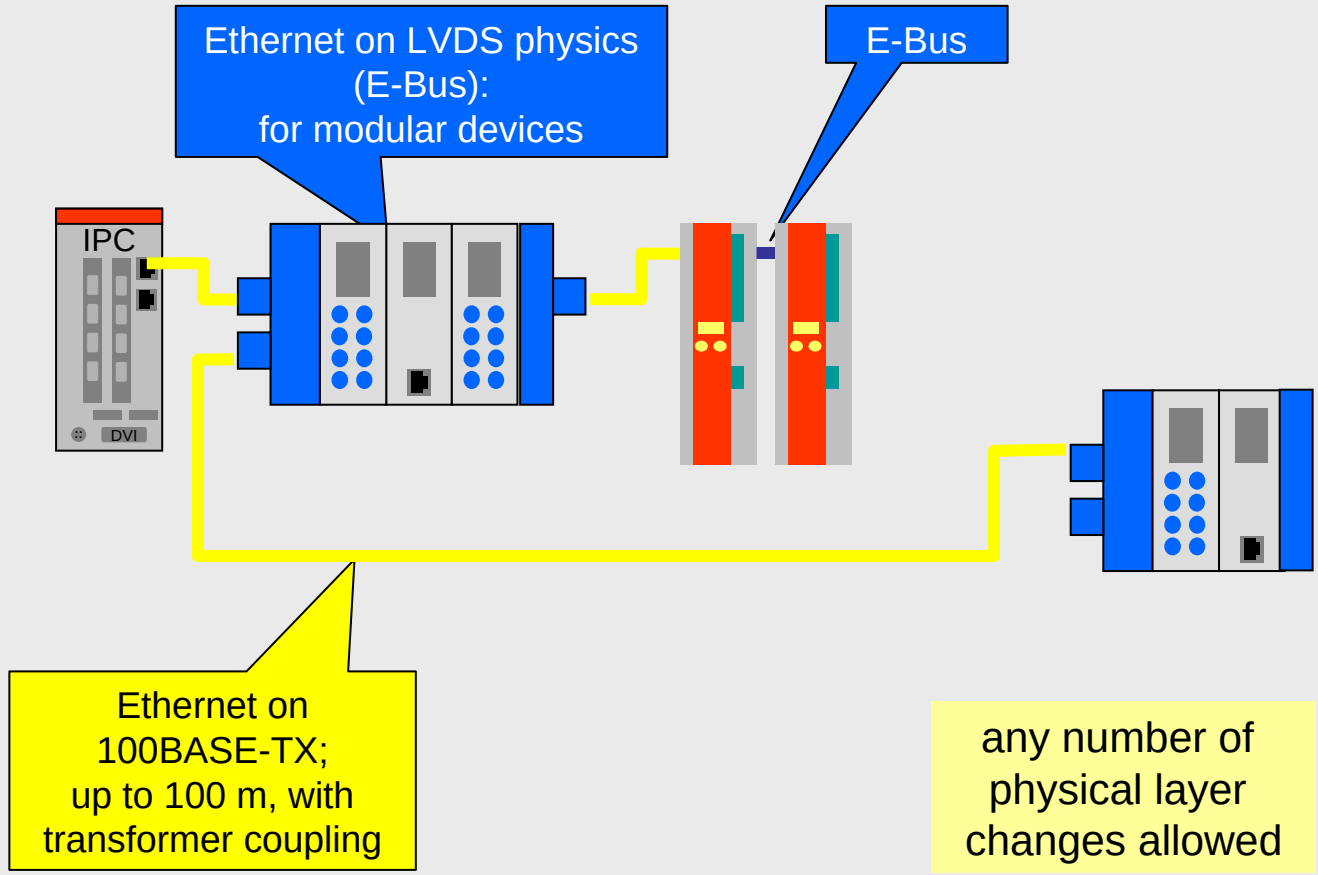
EtherCAT Slave Controller Board



* Post stamp design, not cost and space optimized

EtherCAT Physical Layers

- On cables: 100BaseTX or 100BaseFx
- Device internal: E-Bus (LVDS)

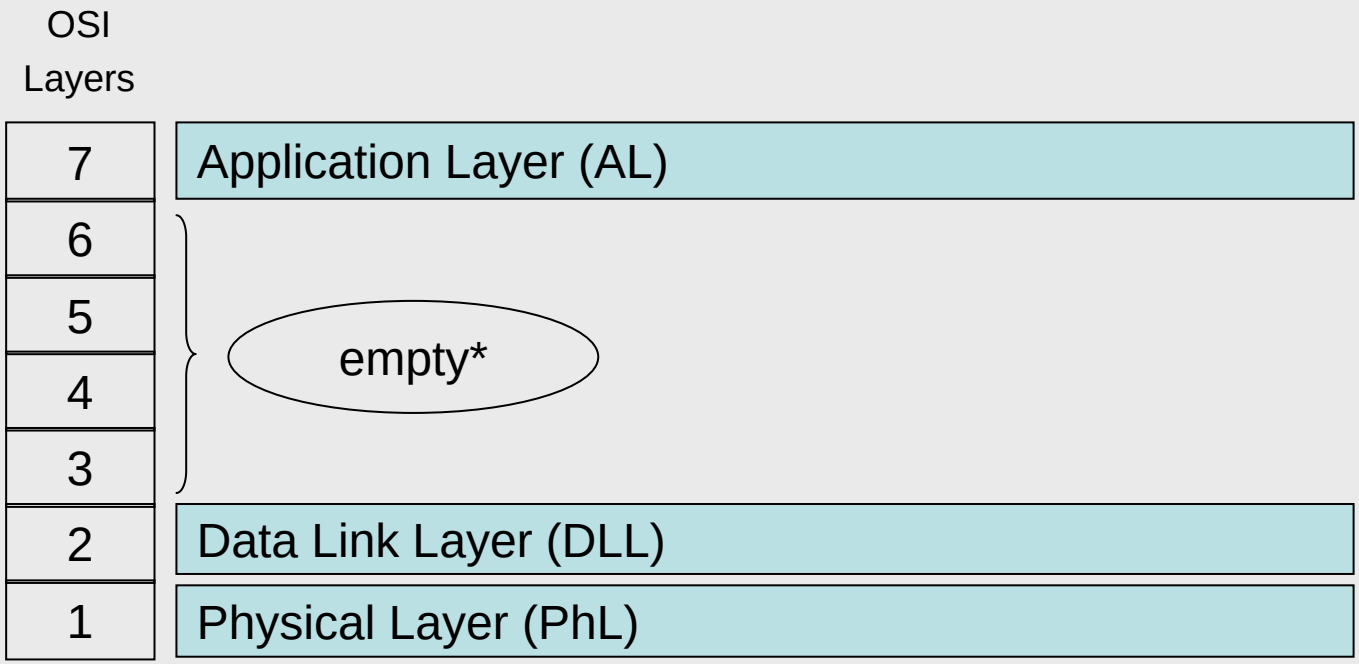


Physical Layer

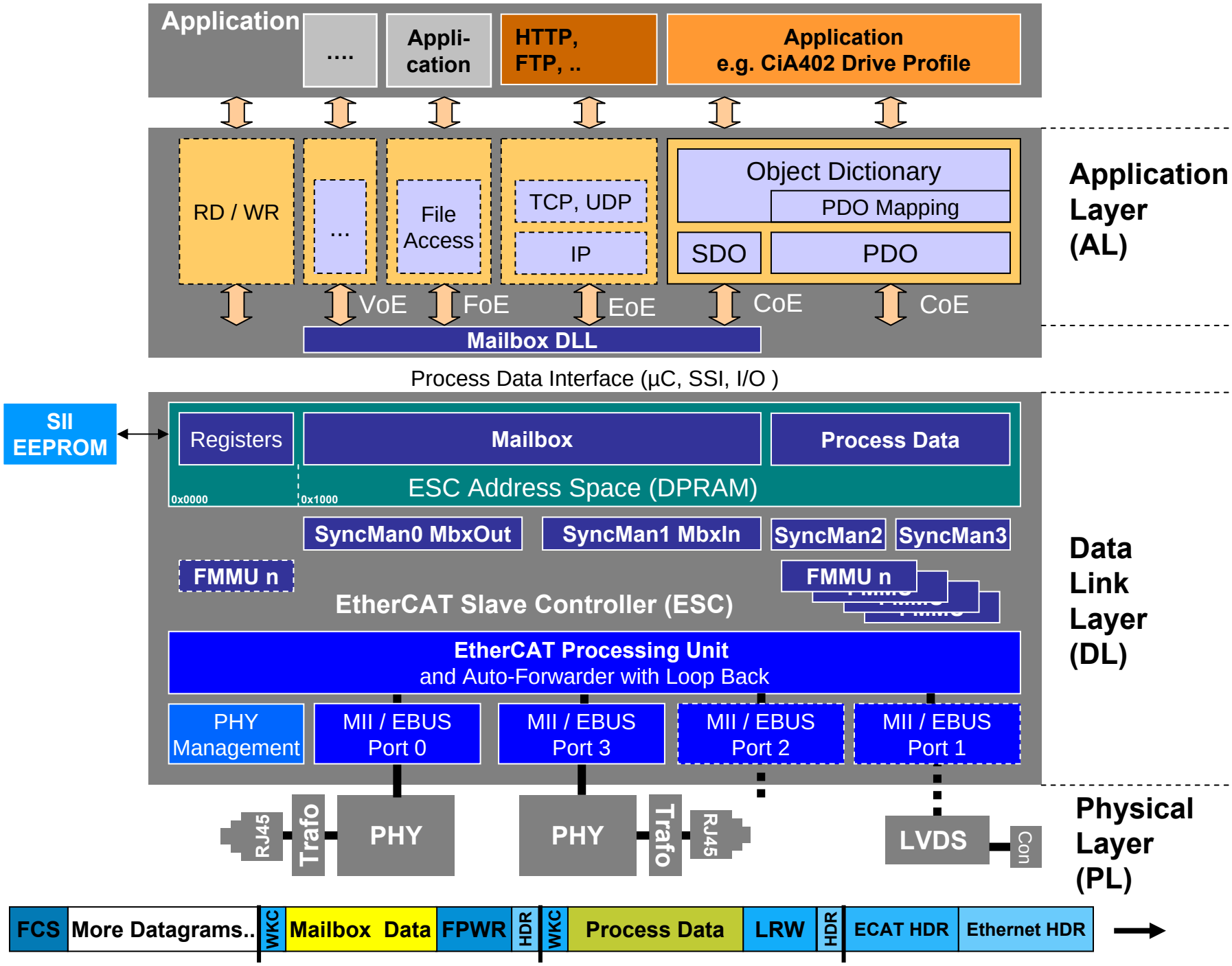
- 100 BASE-TX
  - Most popular physical layer for Fast Ethernet
  - Shielded twisted pair (STP) with 2 pairs of wires
  - Cable categories CAT5, 6, 7 can be used
  - RJ45 connector standard, M12 connector for IP67
  - PHY Support for auto negotiation and auto crossover recommended
- 100 BASE-FX
  - All media options possible
  - Simple solution for TX-to-FX converter

- E-BUS
  - Interface for low cost backplane applications
  - Widely used LVDS (Low Voltage Differential Signaling) adopted
  - Use Manchester Bit Coding
  - LVDS: Low Voltage Differential Signaling according to ANSI/TIA/EIA-644, also used in IEEE 802.3ae (10Gigabit Ethernet)

ISO/OSI Layer Model



- *** „empty“ means that the layer behavior exists, but is not shown explicitly**


Application
....
Appli-
cation
HTTP,
FTP, ..
Application
e.g. CiA402 Drive Profile
RD / WR
...
File
Access
TCP, UDP
IP
Object Dictionary
PDO Mapping
SDO
PDO
VoE
FoE
EoE
CoE
CoE
Mailbox DLL
Application
Layer
(AL)
Process Data Interface (µC, SSI, I/O)
SII
EEPROM
Registers
Mailbox
Process Data
ESC Address Space (DPRAM)
0x0000
0x1000
SyncMan0 MbxOut
SyncMan1 MbxIn
SyncMan2
SyncMan3
FMMU n
FMMU n
EtherCAT Slave Controller (ESC)
EtherCAT Processing Unit
and Auto-Forwarder with Loop Back
PHY
Management
MII / EBUS
Port 0
MII / EBUS
Port 3
MII / EBUS
Port 2
MII / EBUS
Port 1
Data
Link
Layer
(DL)
RJ45
Trafo
PHY
PHY
Trafo
RJ45
LVDS
Con
Physical
Layer
(PL)
FCS
More Datagrams..
WKC
Mailbox Data
FPWR
HDR
WKC
Process Data
LRW
HDR
ECAT HDR
Ethernet HDR

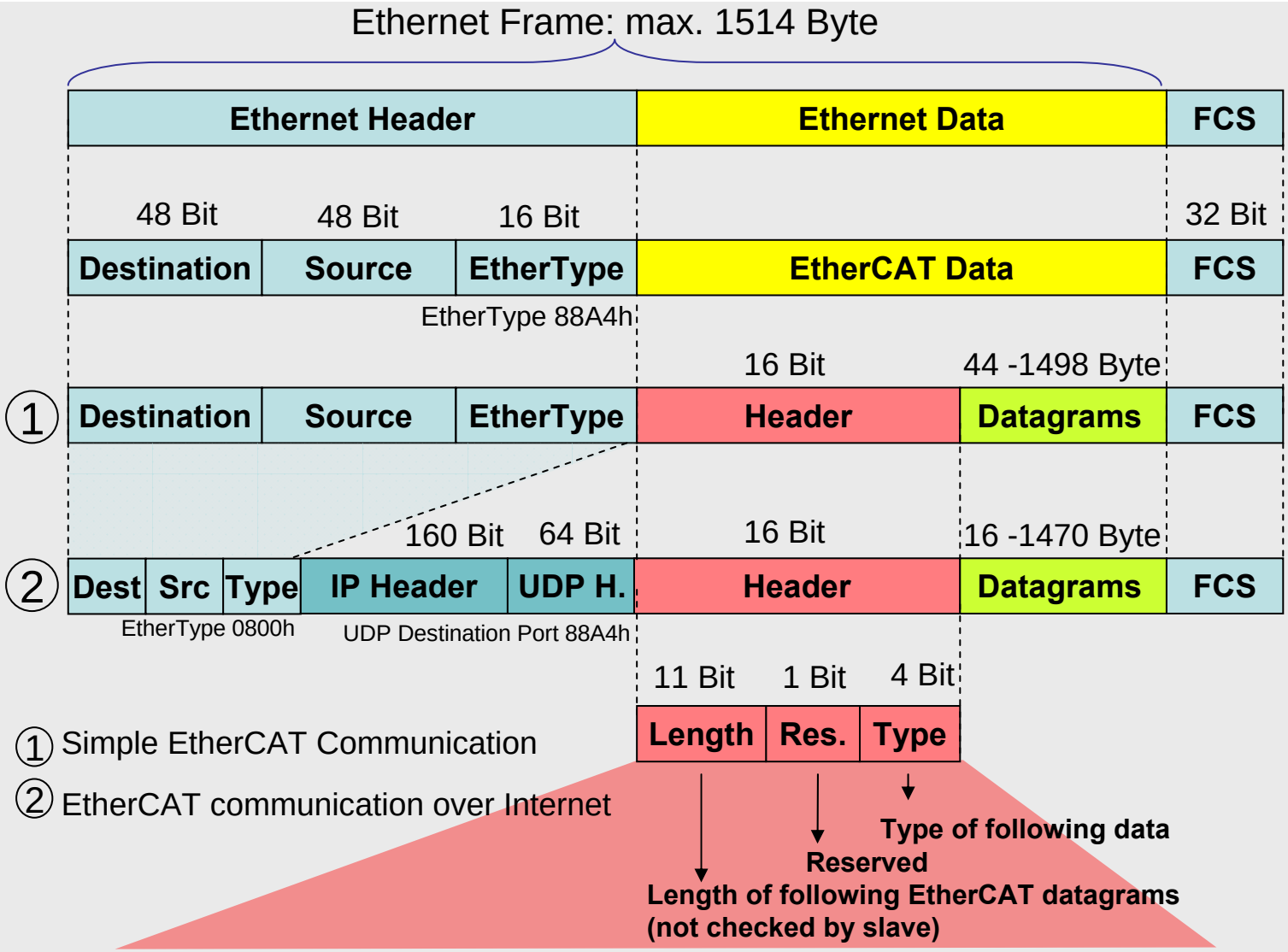
Purpose of Data Link Layer

- Data Link Layer links Physical and Application Layer
- Data Link Layer takes care of the underlying communication infrastructure
  - Link Control
  - Access to Transceivers (PHY)
  - Addressing
  - Slave Controller configuration
  - EEPROM access
  - SyncManager configuration and management
  - FMMU configuration and management
  - Process Data Interface configuration
  - Distributed Clock
  - Set Up AL State Machine interactions

Data Link Layer – Overview

- Standard IEEE 802.3 Ethernet Frame
  - No special requirements for the master
  - Use of standard Ethernet infrastructure
- IEEE Registered EtherType: 88A4h
  - Optimized frame overhead
  - IP stack not required
  - Simple master implementation
- Additionally over UDP
  (IANA registered Port 88A4h)
  - EtherCAT communication over the Internet possible
  - Using of standard sockets
- Frame processing at Slave side
  - EtherCAT Slave Controller processes the frame in hardware
- Communication Performance independent from processor power
  - no time critical reaction at slave side in software

Ethernet / EtherCAT Frame Structure

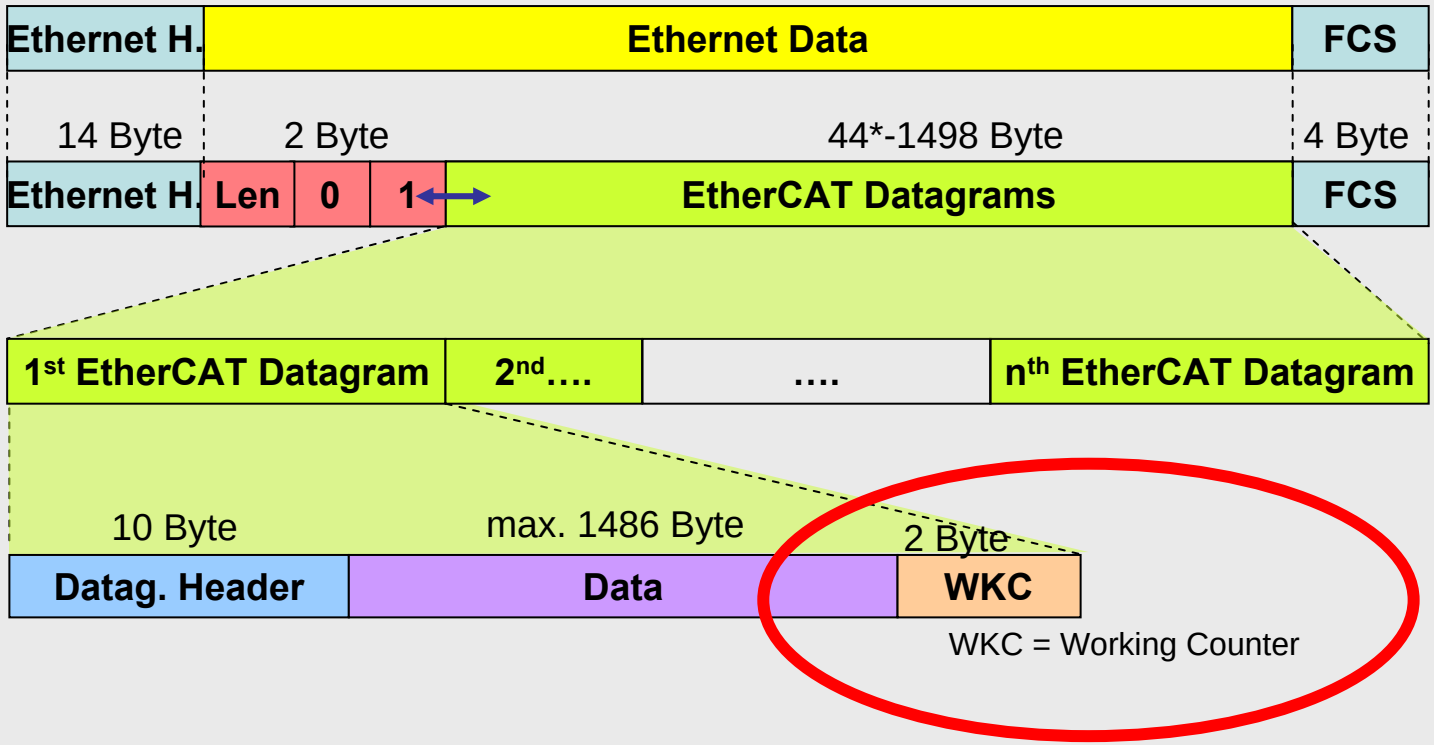


EtherCAT Frame Header

Type	Meaning
0:	Reserved
1:	EtherCAT Datagram (s)
	the only type that is evaluated by the ESC
2,3:	Reserved
4:	Network Variables
5:	Mailbox over IP
6-15:	Reserved for future use

EtherCAT Datagrams

* add 1-32 padding bytes if Ethernet frame is less than 64

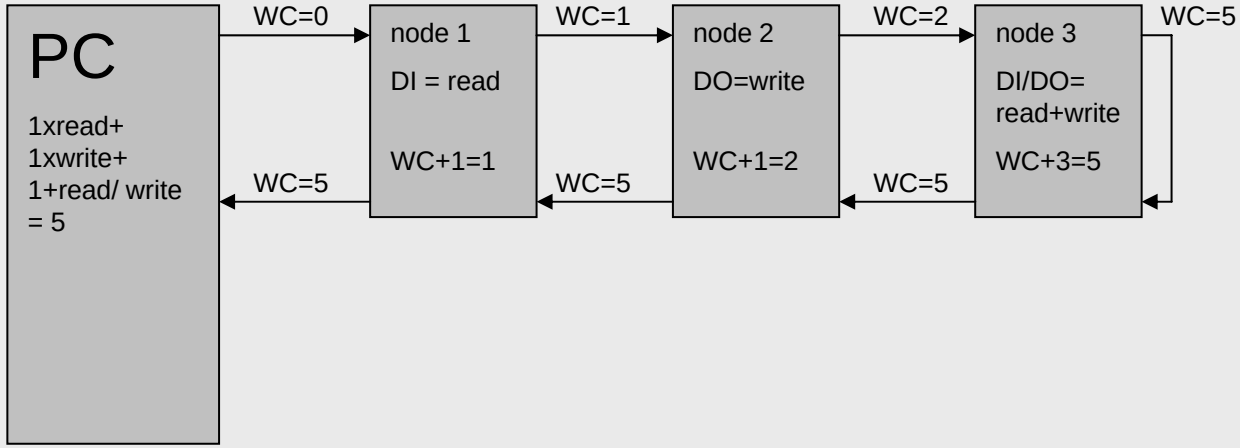


Working Counter Details

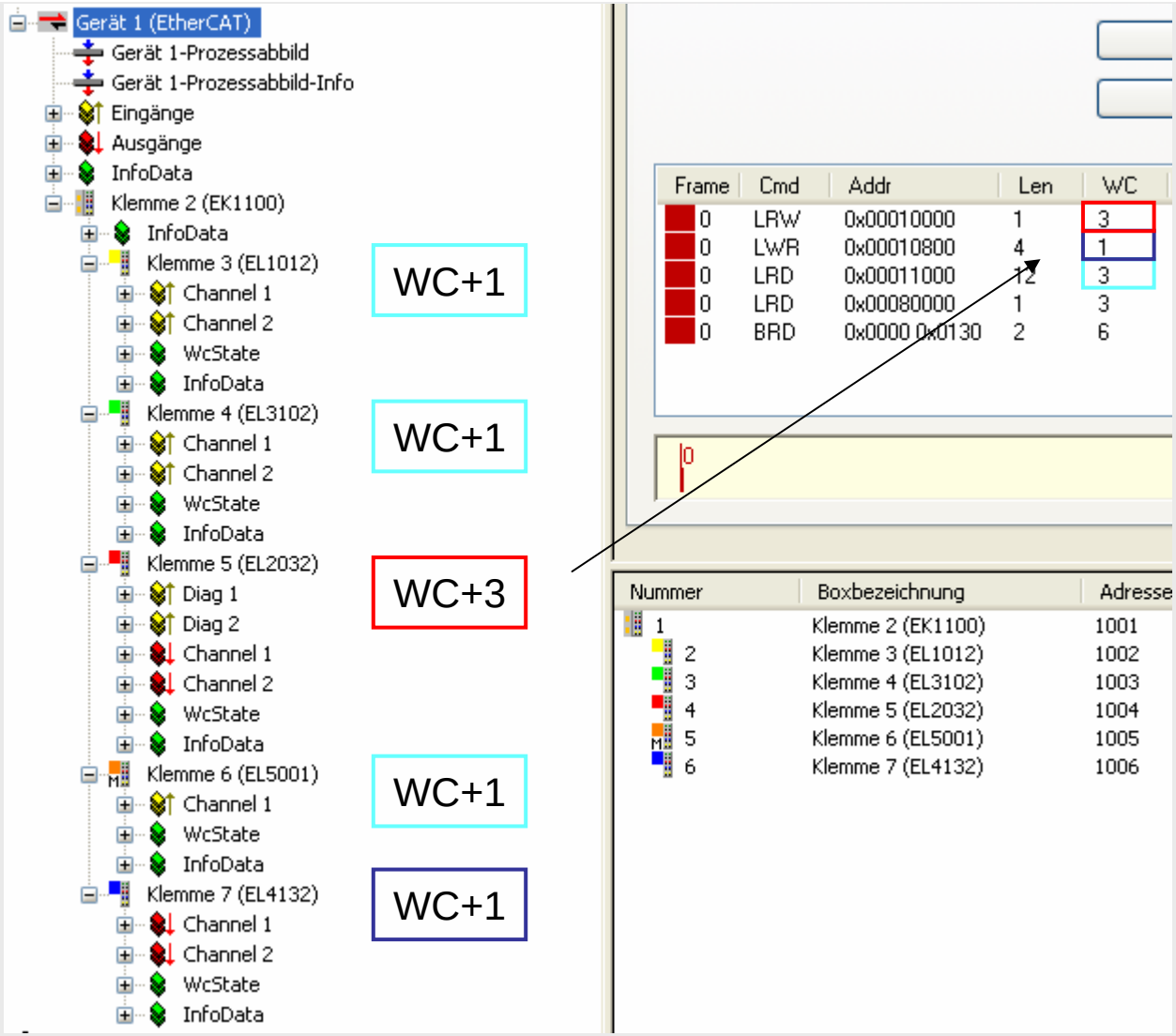
- EtherCAT Datagram ends with a 16 Bit Working Counter
- Working Counter counts the number of interactions of devices addressed by an EtherCAT Datagram
- EtherCAT Slave Controller increments the Working Counter in hardware – if the controller is addressed and the addressed memory is accessible (Sync Manager)
- Each Datagram should have an expected Working counter value – calculated by the configuration tool
- The Master checks the valid processing of EtherCAT Datagrams by comparing the Working Counter with the expected value
- Special case: RW addressing methods will increment WKC by 2 for write access and by 1 for read access

Working Counter Example

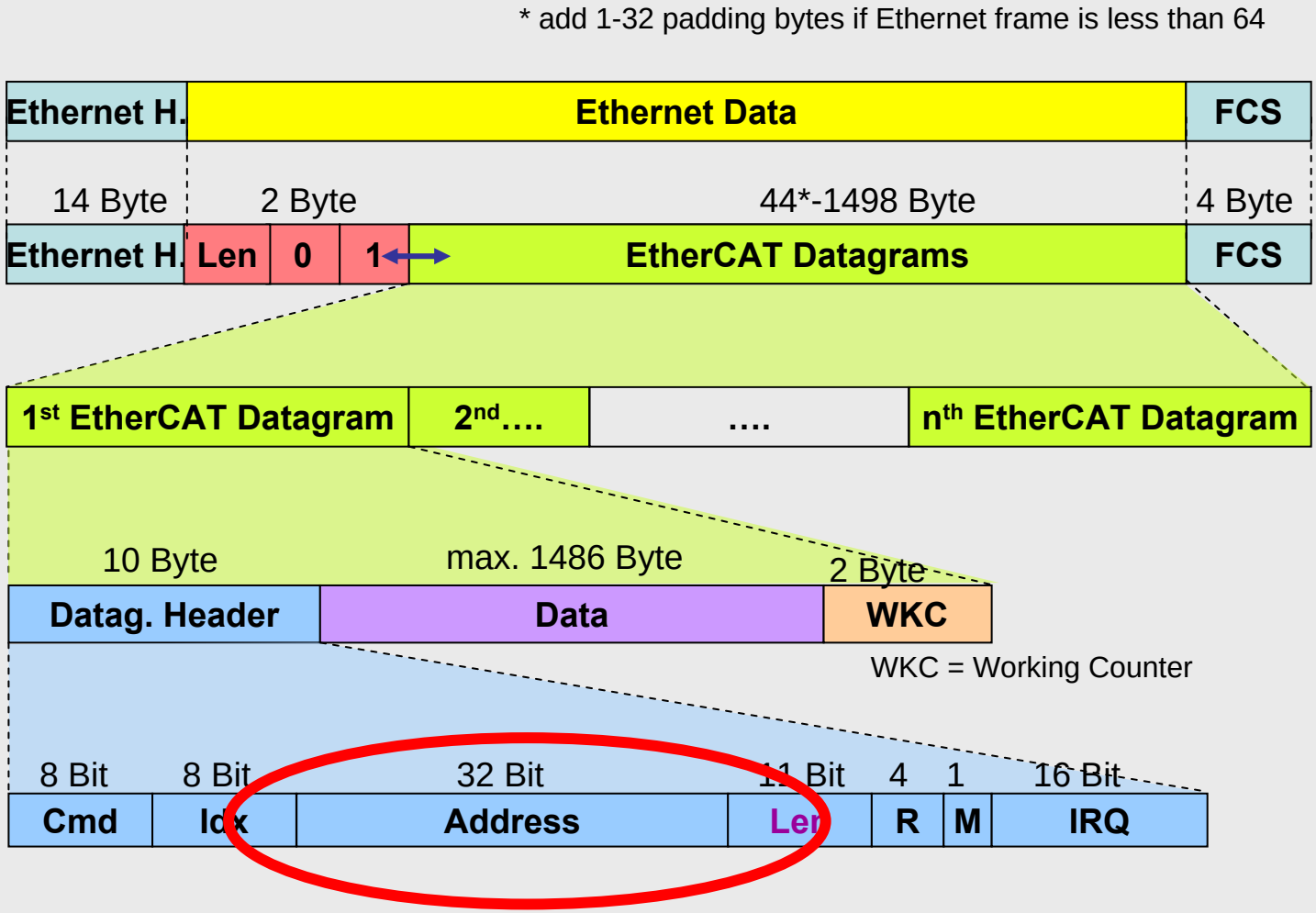
- WKC valid: data of this datagram was written to and read from all addressed devices
- WKC invalid: memory of one or more devices was not accessible



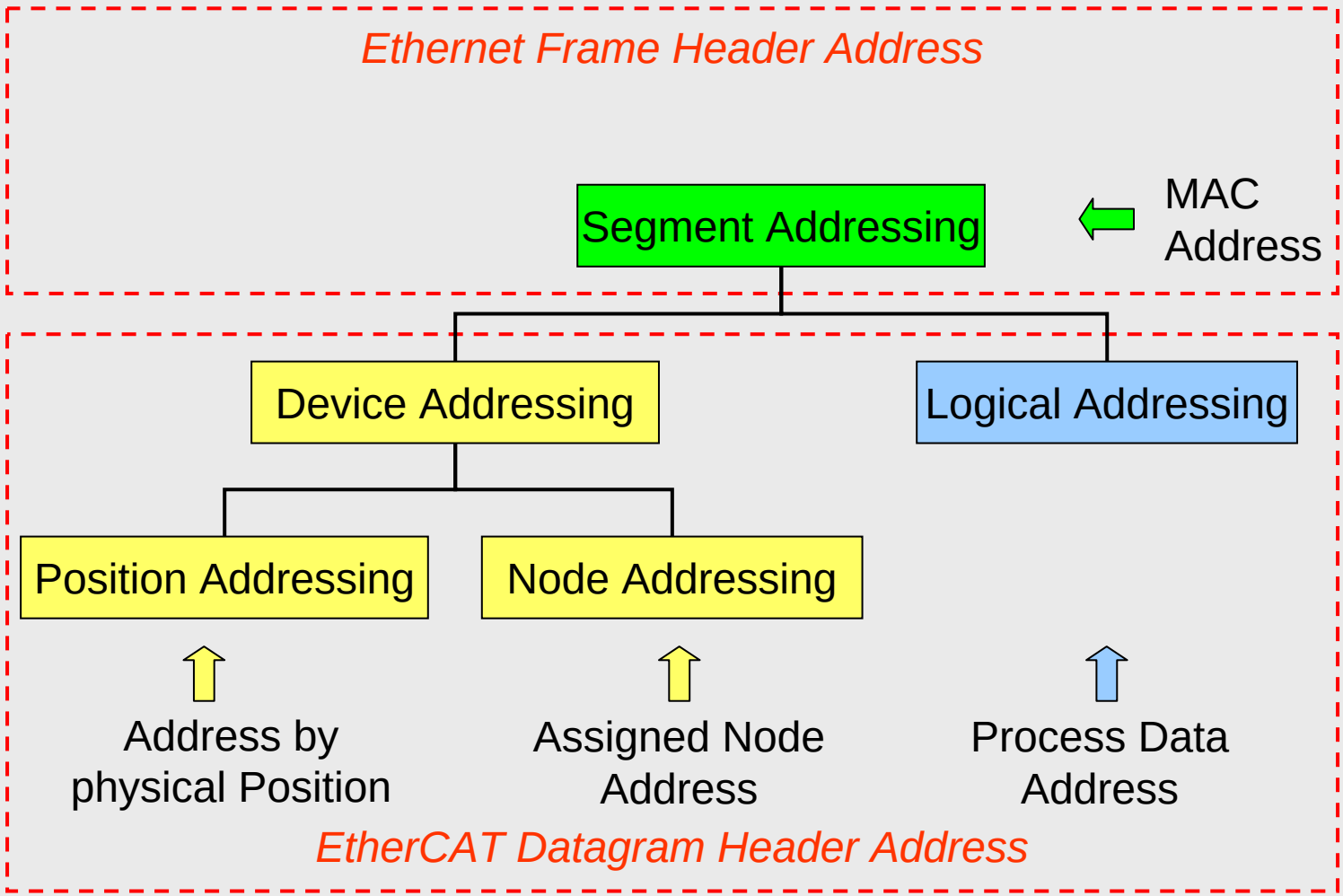
Working Counter Example



EtherCAT Addressing

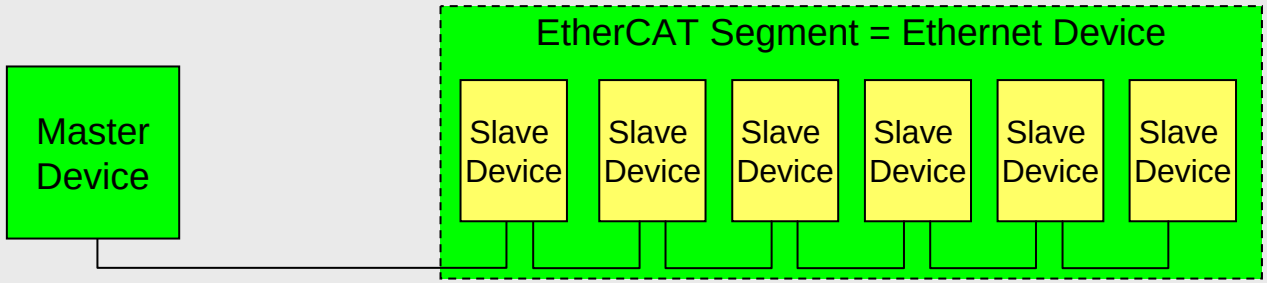


EtherCAT Addressing Overview

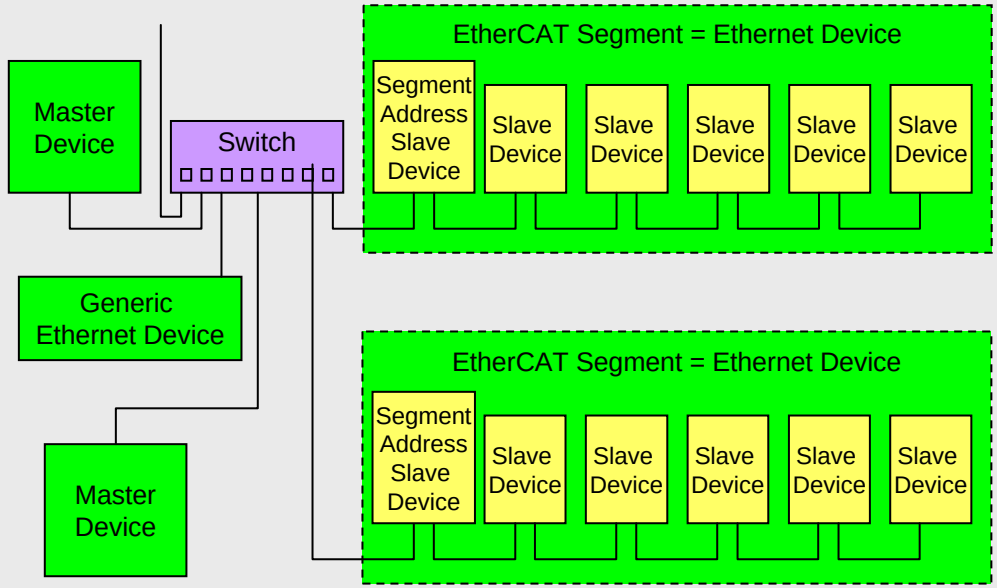


Segment Addressing by MAC Address

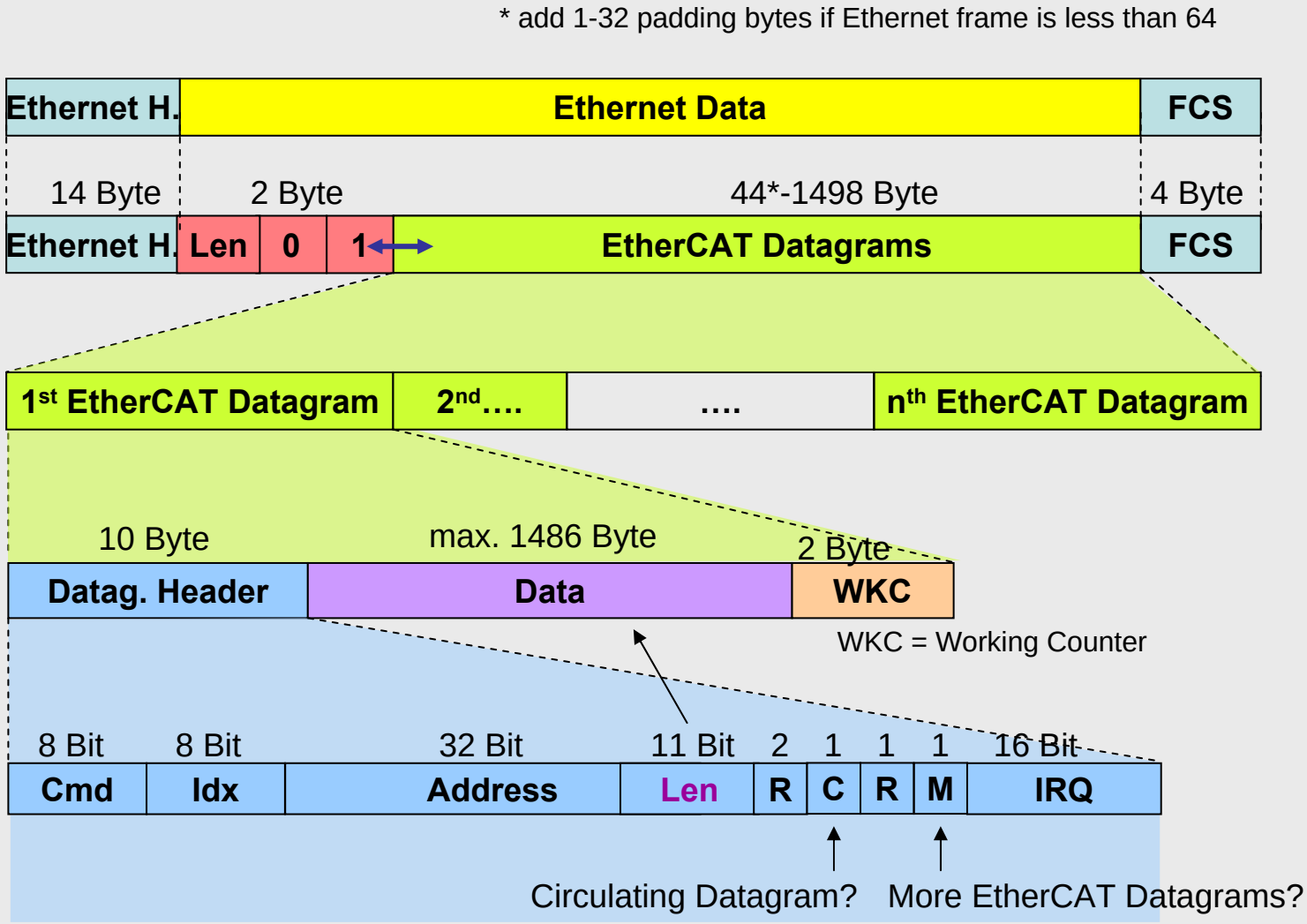
- Direct mode (no switch): broadcast MAC Address



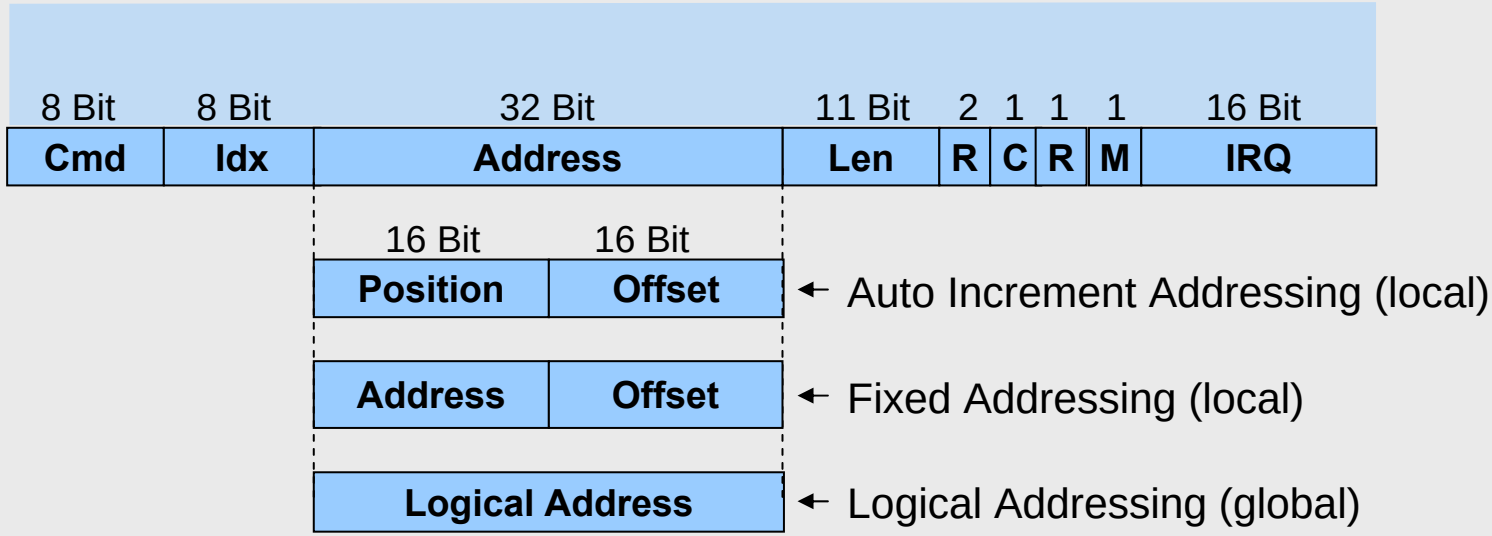
- Open mode: One MAC-Address for an EtherCAT Segment



EtherCAT Datagram Header Address



Address Field



- 32 Bit address space
  - used for 16 bit device addressing (65.535 devices possible)
    and 16 bit for addressing local memory space of device (max. 64kByte)

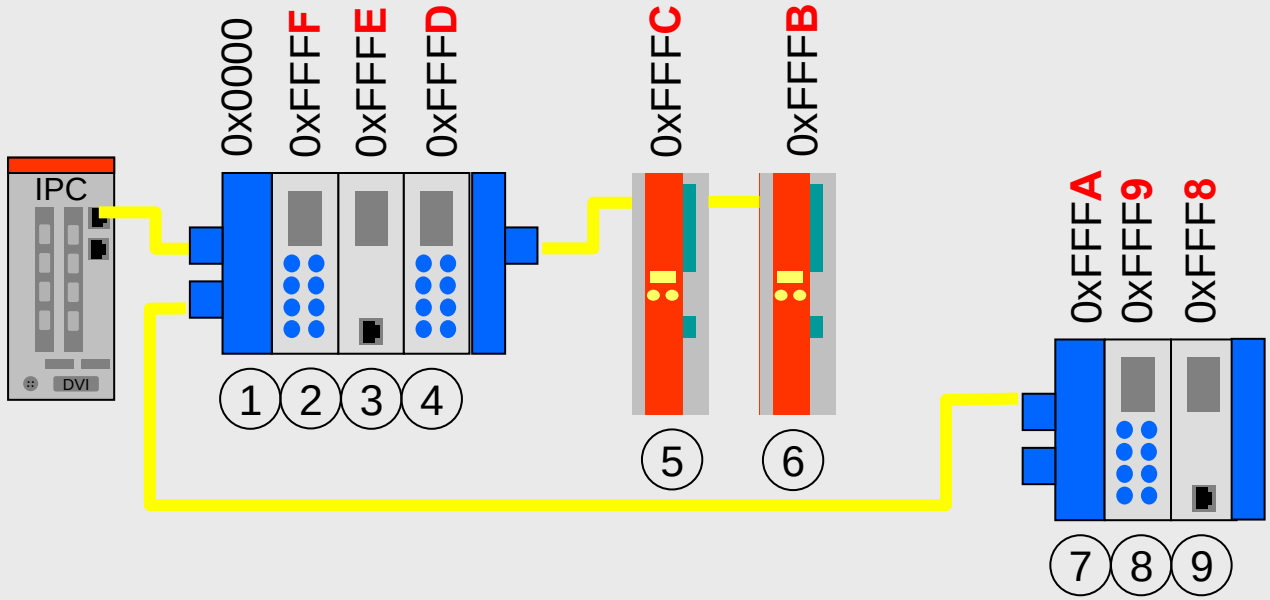
  or

  - 32 bit logical addressing

Auto Increment Addressing

16 Bit	16 Bit
Position	Offset

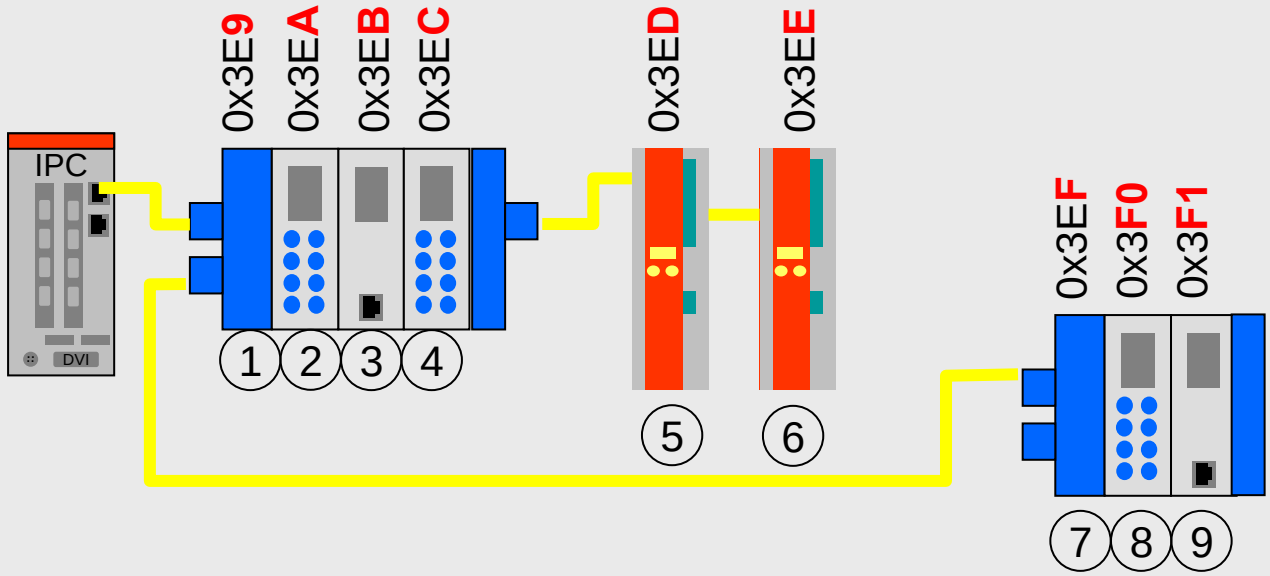
- Negative Auto Increment Address for every slave depending on position (16 bit)
- Slave which reads address == 0x0000 is addressed
- Every slave increments address by 1
- Offset addresses local memory space of device
- Usually used during scan of hardware configuration



Fixed Addressing

16 Bit	16 Bit
Address	Offset

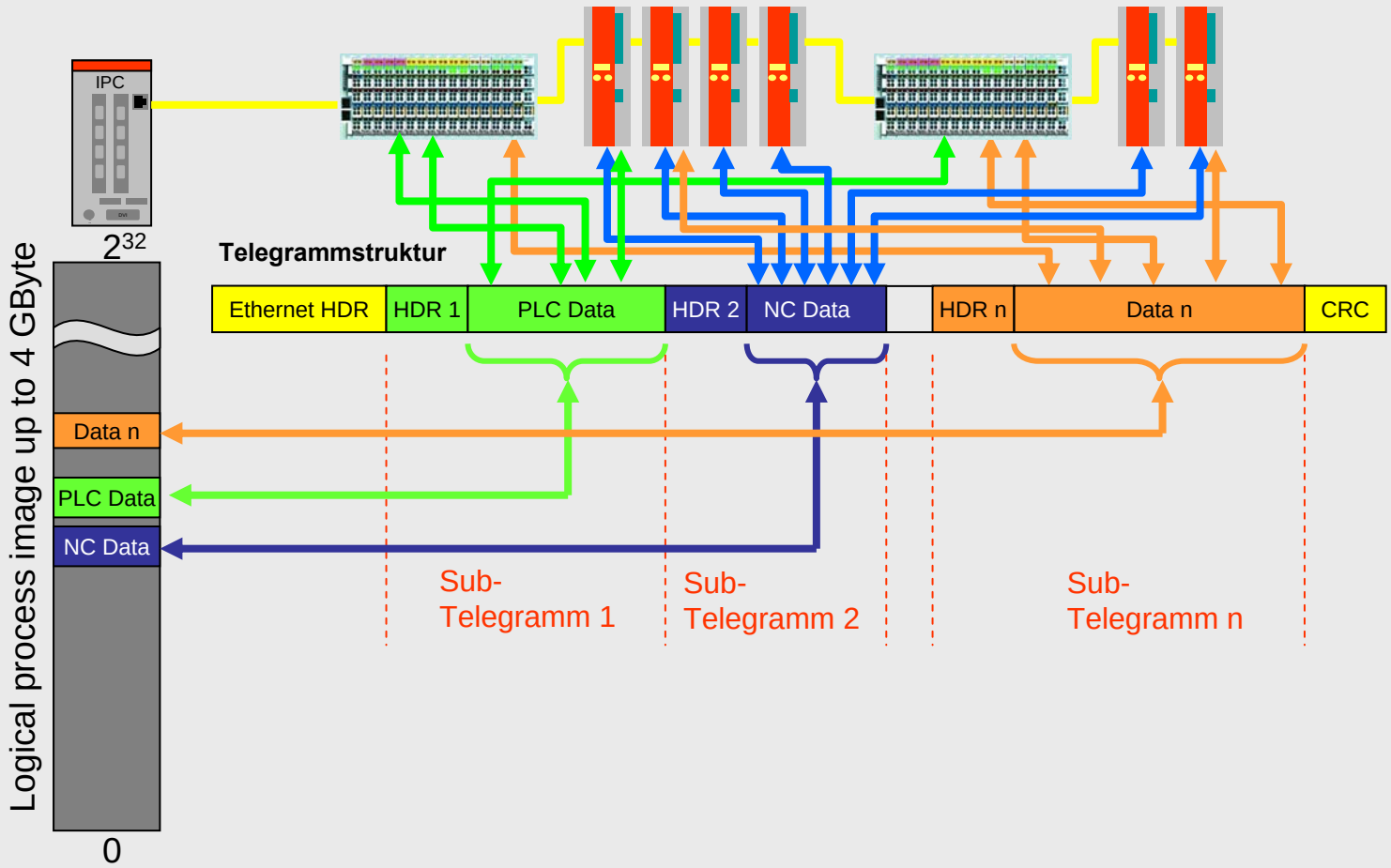
- Every Slave has a fixed address (16 bit)
- Usually assigned during hardware configuration scan
- Independent from slave position
- Fixed address lost after power loss



Logical Addressing

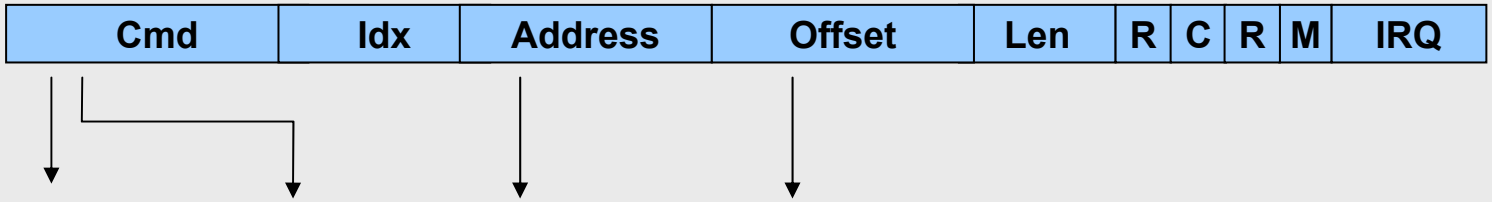
32 Bit
Position

- Slave reads from/ writes its data into the 4 GByte great Ethernet frame (fragmented)



EtherCAT Commands

- Different commands to optimize reading and writing for all access methods within a Fieldbus communication system

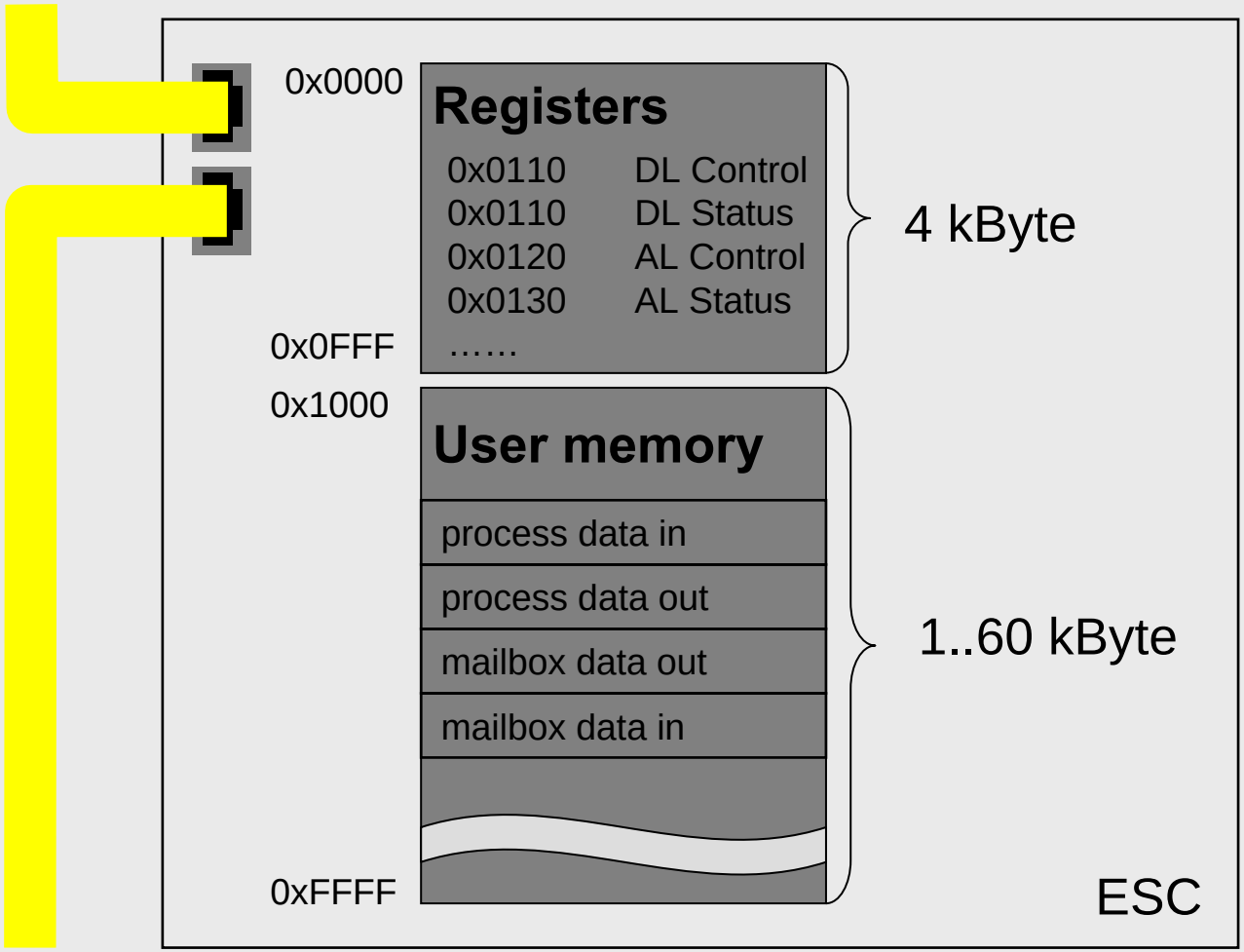


Cmd Type	Access	Address	Offset	Comment
NOP				No Operation
Auto Increment	R, W, RW, RMW	Position (increments)	Local Memory Address	Position value 0 (at entry) addressed
configured Address	R, W, RW, RMW	Address (configured)	Local Memory Address	Match address value to local address register
Broadcast	R, W, RW	(increments)	Local Memory Address	
Logical	R, W, RW	32 Bit logical address		

EtherCAT Commands

- Broadcast Read
  - Individual Bits of a Byte will be added with a bitwise OR operation between incoming data and local data
- Read Write Actions
  - Exchange of incoming data and local data (exception: Broadcast – see broadcast read)
- Read Multiple Write Actions (RMW)
  - Addressed Station will read the others will write

Local Address Space of ESC



Local Address Space of ESC

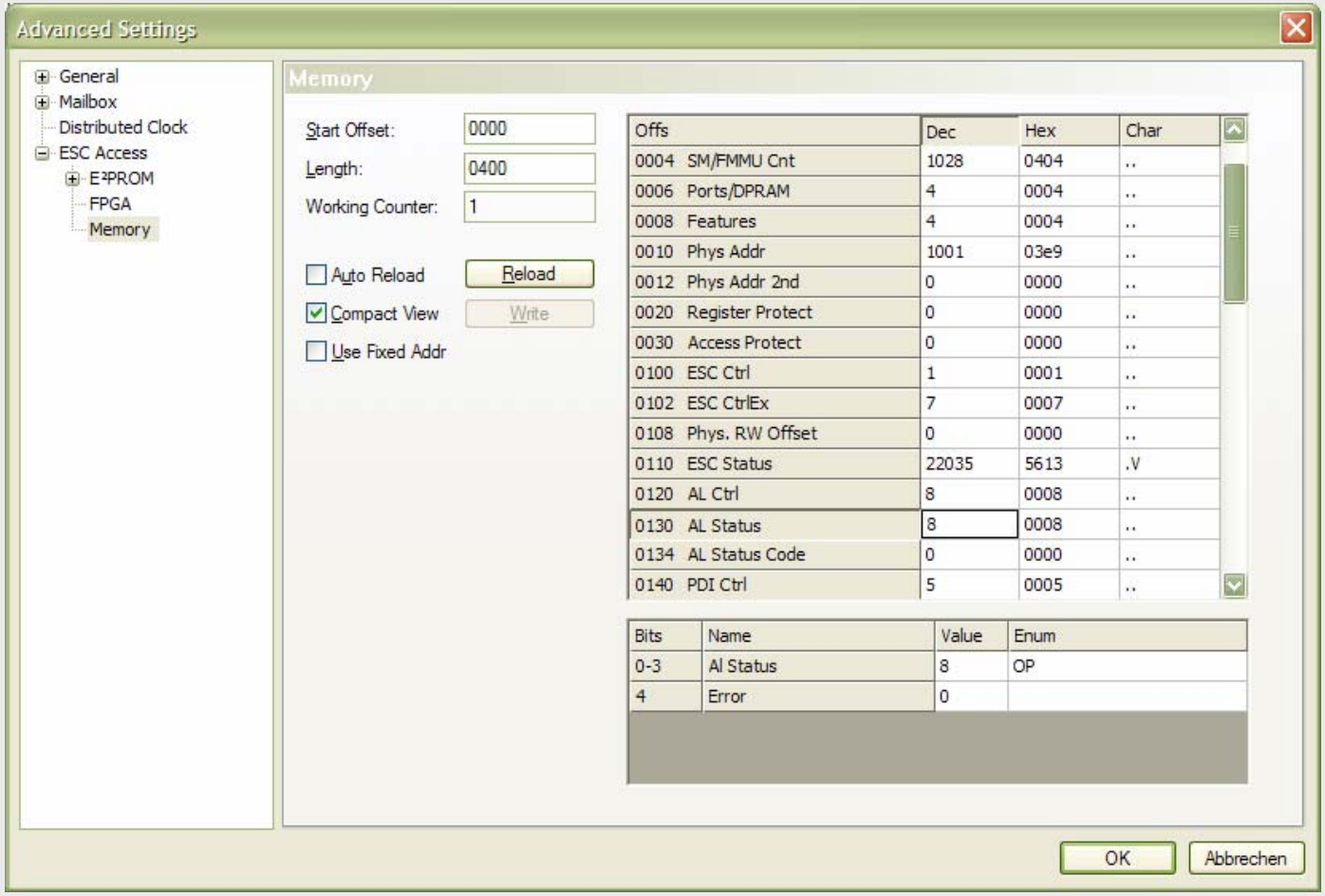
- 64 kByte address space
- Divided into registers and dual ported RAM (DPRAM)
- First 4 kByte are reserved for registers
- DPRAM starts at 1000h
- DPRAM size depends on Slave Controller implementation (up to 60 kByte, 4kByte in actual FPGA implementation)
- Addressing of registers and DPRAM same
- Register Write is different –
  shadow Register for all Registers integrated
  DPRAM write is not shadowed

Register of EtherCAT Slave Controller

- First 1000h bytes (4 kbytes) of local address space
- Read access for both sides (EtherCAT and application)
- Write access from EtherCAT for most of the registers
  - Master has to configure the Slave Controller
    - No address settings needed
    - FMMU and Sync Manager configuration can be optimized for available bandwidth and cycle times
  - Exceptions that are writable from the application side:
    - AL Status Register, AL Status Code Register, AL Event Mask Register, Sync Manager Disable Registers, AL Identification Registers
- Process Data Interface (PDI) register initialized from Slave Information interface (Serial EEPROM)

Register

- Registers might be monitored via configuration tool



Register – ESC Data Sheet

- Register description for every ESC (FPGA/ ASIC)
  - DL Information, DL Control, DL Status, DL Address
  - AL Control, AL Status, AL Event
  - SyncManager + FMMU configuration
  - Distributed Clocks
  - Slave Information interface (Serial EEPROM)

Address	Length (Byte)	Description	EtherCAT Access	PDI Access
0x0100:0x0103	4	DL Control	r/w	r/-
0x0108:0x0109	2	Physical Read/Write Offset	r/w	r/-
0x0110:0x0111	2	DL Status		r/-
0x0120:0x0121	2	AL Control		
0x0130:0x0131	2	AL Status		
0x0134:0x0135	2	AL Status Code	r/-	r/w
0x0140:0x0141	2	PDI Control		

Enable Disable Ports

Control of the device state machine

Status of the device state machine

Error Code

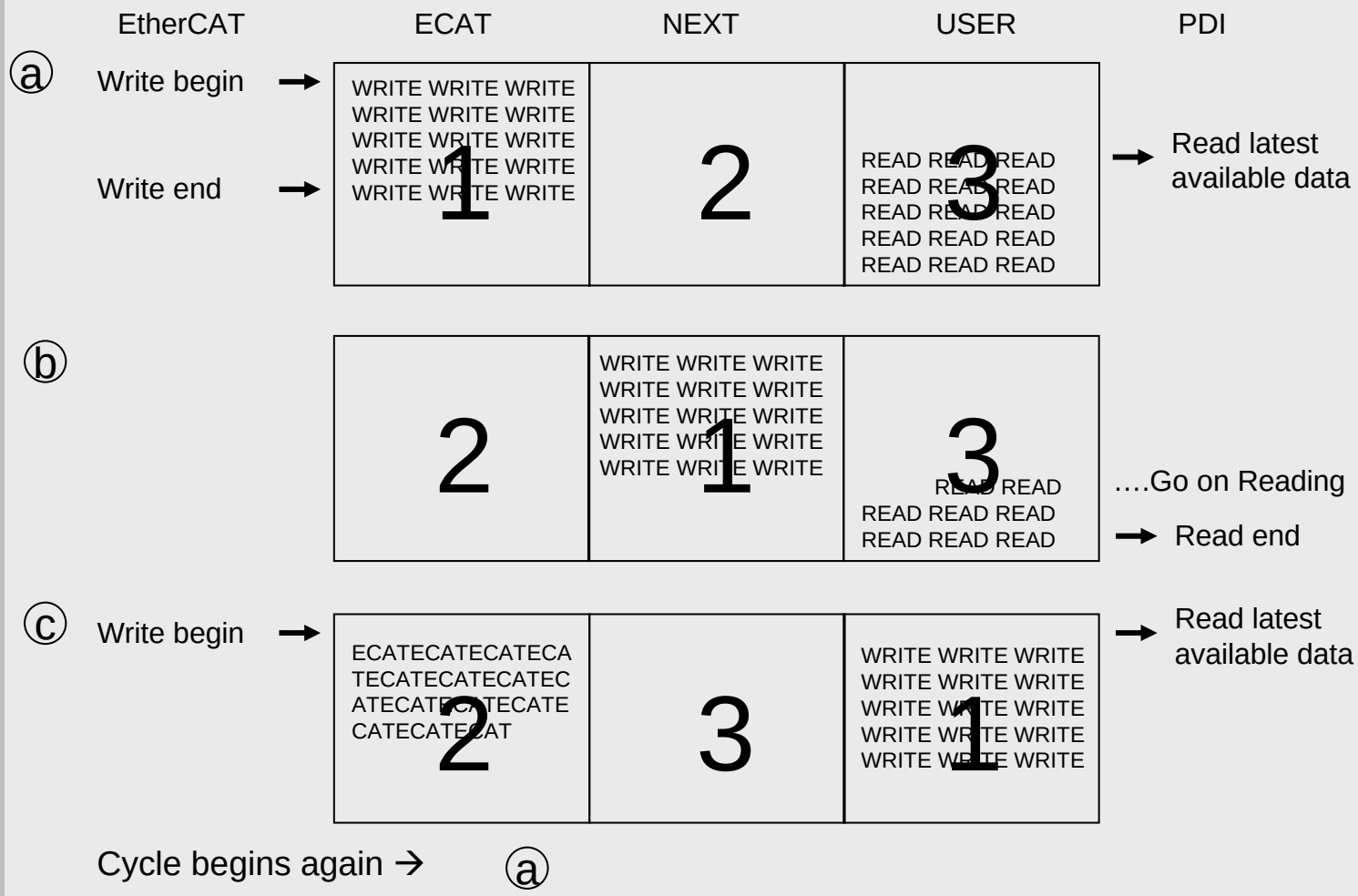
32 Bit I/0, SPI, µC-Interface

SyncManager Overview

- SyncManager protects a DPRAM section from simultaneous access → data consistency
- Mailbox Type
  - 1 buffer SyncManager supports handshake
  - Data overflow protection
  - Writing side must write before reading side can read
  - Reading side must read before writing side can write again
- Buffered Type
  - 3 buffer SyncManager guarantees consistent data delivery and access to the newest data any time
  - Always a free buffer to write
  - Always a consistent buffer to read (except before the first writing)
  - Usually used for process data communication
- Up to 16 independent SyncManger channels possible
- The SyncManager configuration registers start at address 0x0800

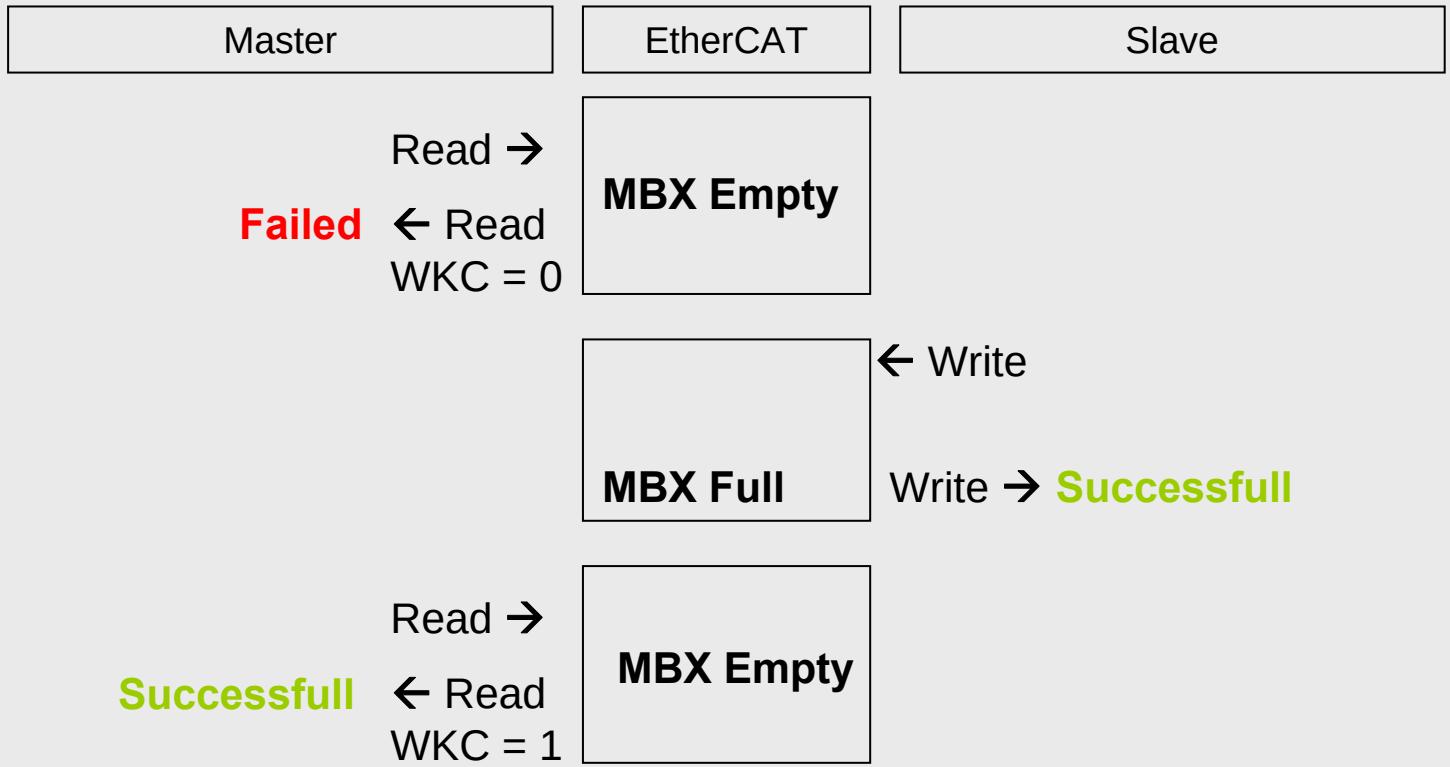
Buffered Type (3 buffers) Write Example

- Characteristic: Data always available for both sides
- Requires 3 (consecutive) memory areas

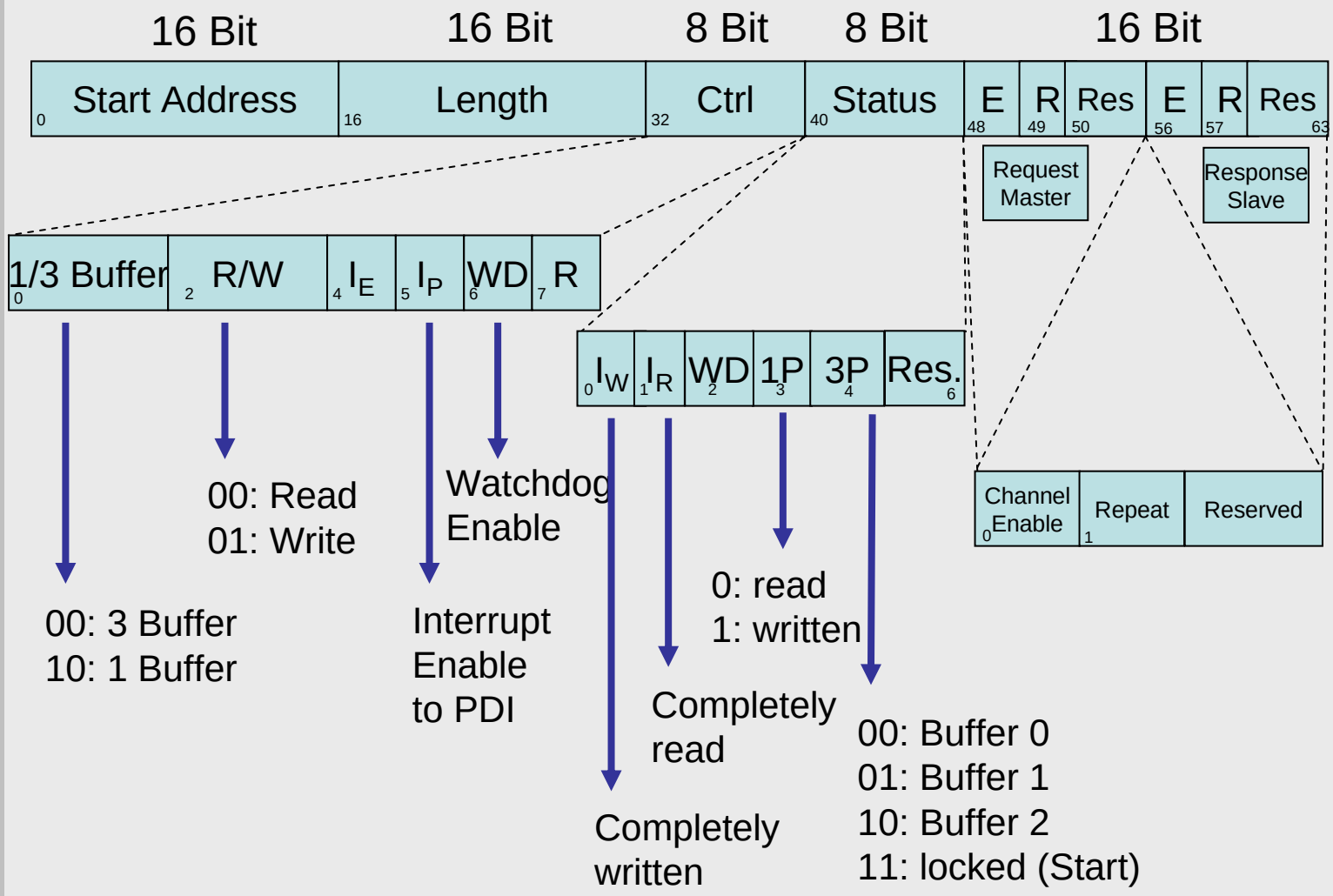


Mailbox Type (1 buffer) Read Example

- Allows handshake Communication
- Useful for non-Process Data
- Handshake mechanism – one side has control



SyncManager channel configuration registers



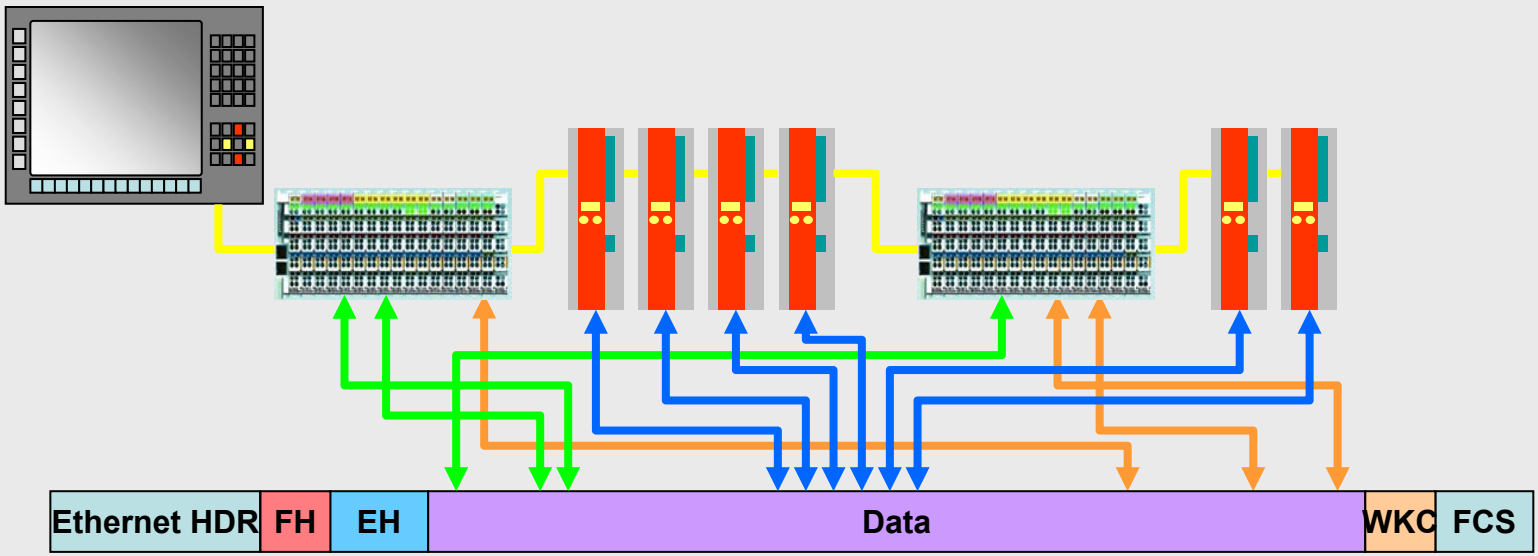
Fieldbus Memory Management Unit (FMMU)

- Maps a section of the local address space into the global address space and vice versa
- Read and write access distinguishable
- Bitwise configuration of the memory section possible
- Up to 16 independent FMMU channels possible
  - FMMU configuration registers start at address 0x0600

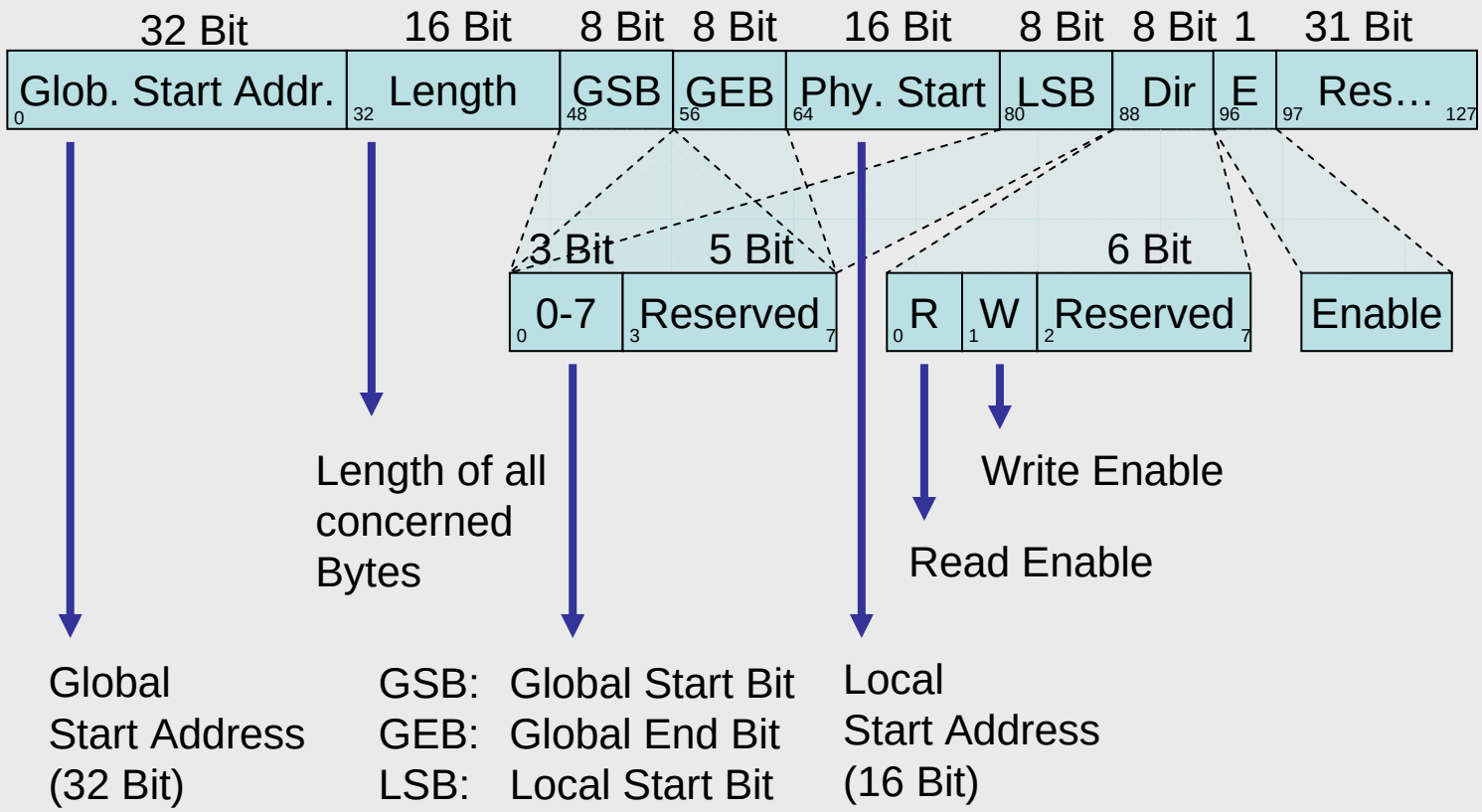
- **Operation samples:**
- Mapping of process data into the global address space
- Mapping of status bits from the register section
  - Access to device specific status information with a minimum overhead – e.g. fill status of a sync manager channel

FMMU Usage for Addressing

- Global address space
- 4 GByte address space
- Mapping to local addresses by Fieldbus Memory Management Units (FMMU)
- All EtherCAT devices can map data in a single EtherCAT Datagram LRW – depending on the FMMU configuration



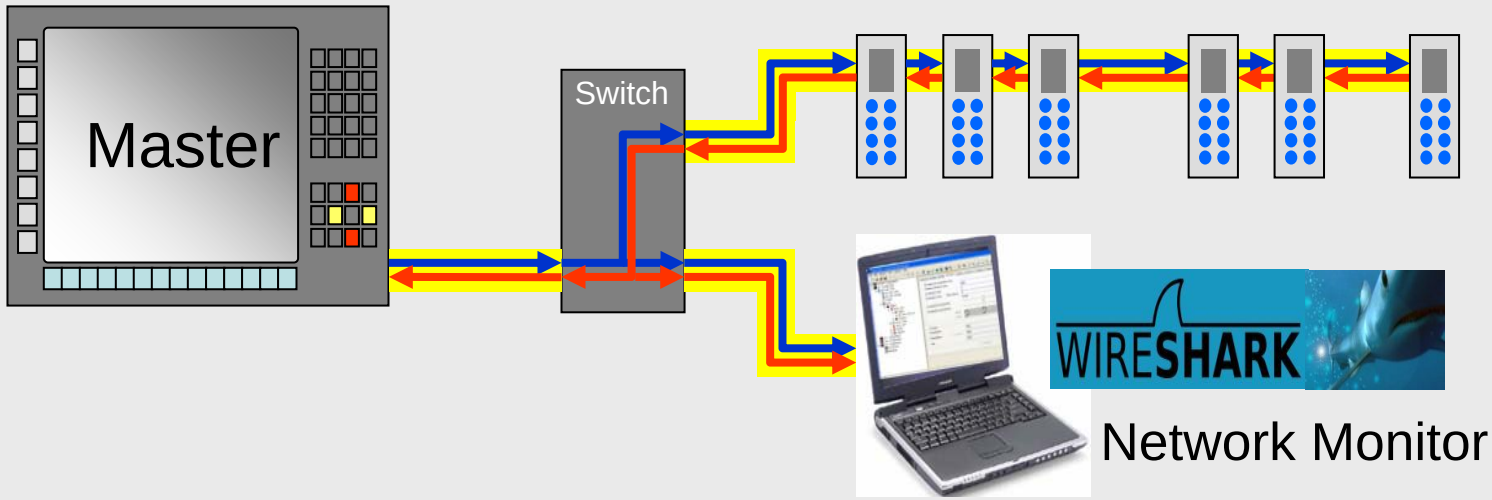
FMMU entity configuration registers



Diagnosis at Data Link – Possible Errors

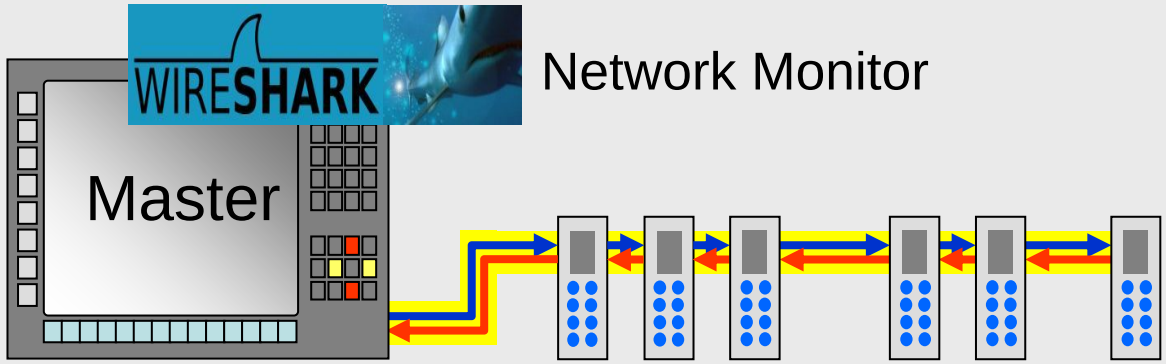
Error situation	Detection
Wrong Station	Check EEPROM, Vendor, Device, SerialNo
Transmission Errors	Check Transmission Error Counter of each station
Change between In and Out Cable	Check Link Status of last device (should have only a single port connected)
Link loss/ Station fault	Check working counter of a Broadcast read
Frame loss	Close loop in the middle Check errors again If no frame loss repeat this in the 2nd half of segment Otherwise repeat this in the first half of segment Repeat this until only a single station remains, which should be the station with problems

Monitoring of EtherCAT Communication



- Masters sends an EtherCAT Frame (broadcast)
  → Monitor gets the first copy (unprocessed)
- Frame returns from EtherCAT Slave Devices
  → Monitor gets the second copy (processed)
- DLL for readable format available on EtherCAT Web site

Monitoring of EtherCAT Communication

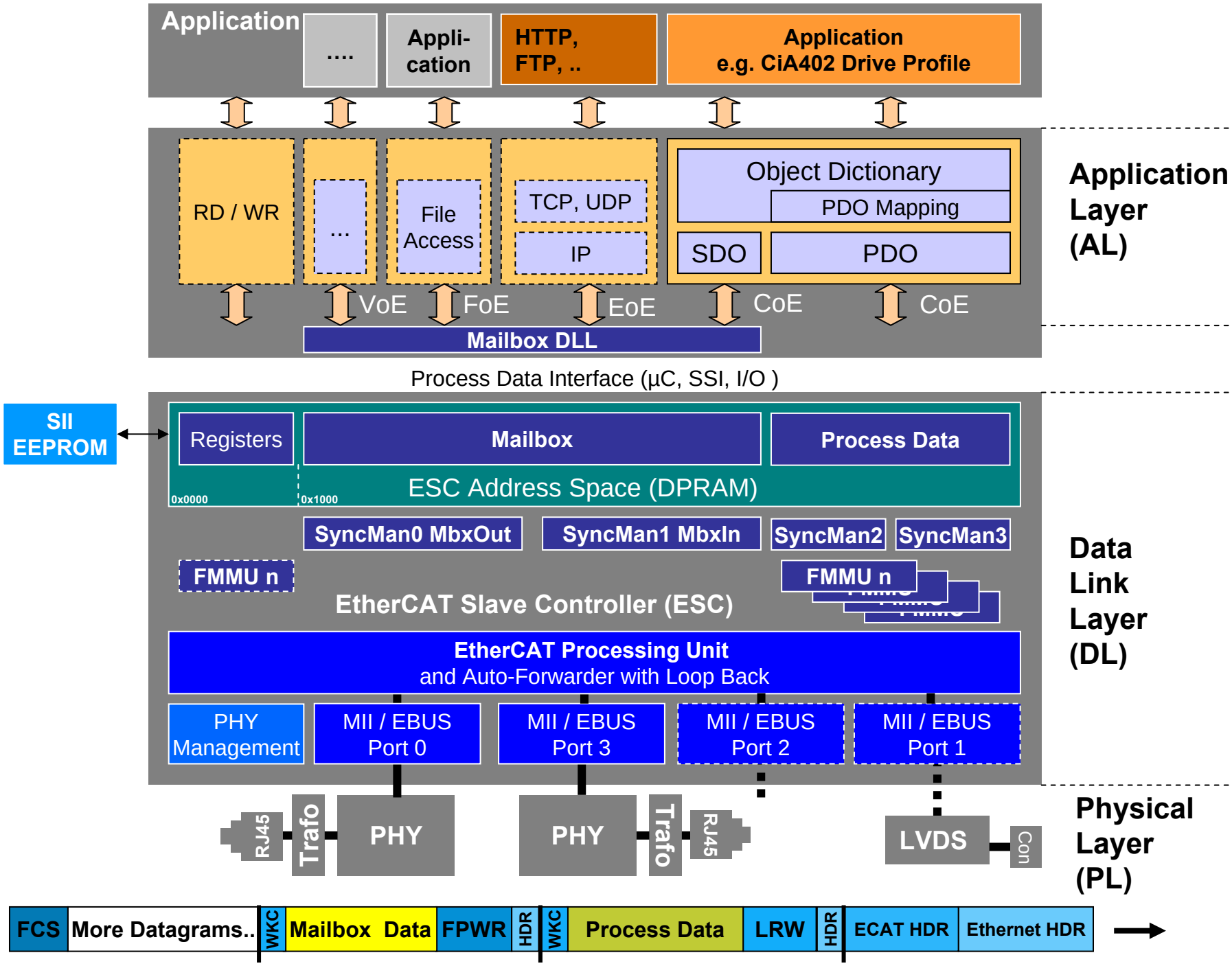


- Masters sends an EtherCAT Frame (broadcast)
  → Monitor gets the first copy (unprocessed)
- Frame returns from EtherCAT Slave Devices
  → Monitor gets the second copy (processed)
- DLL for readable format available on EtherCAT Web site

Attention:
At low cycle times order of frames might be mixed because of timing restrictions within NDIS protocol driver

Purpose of Application Layer (AL)

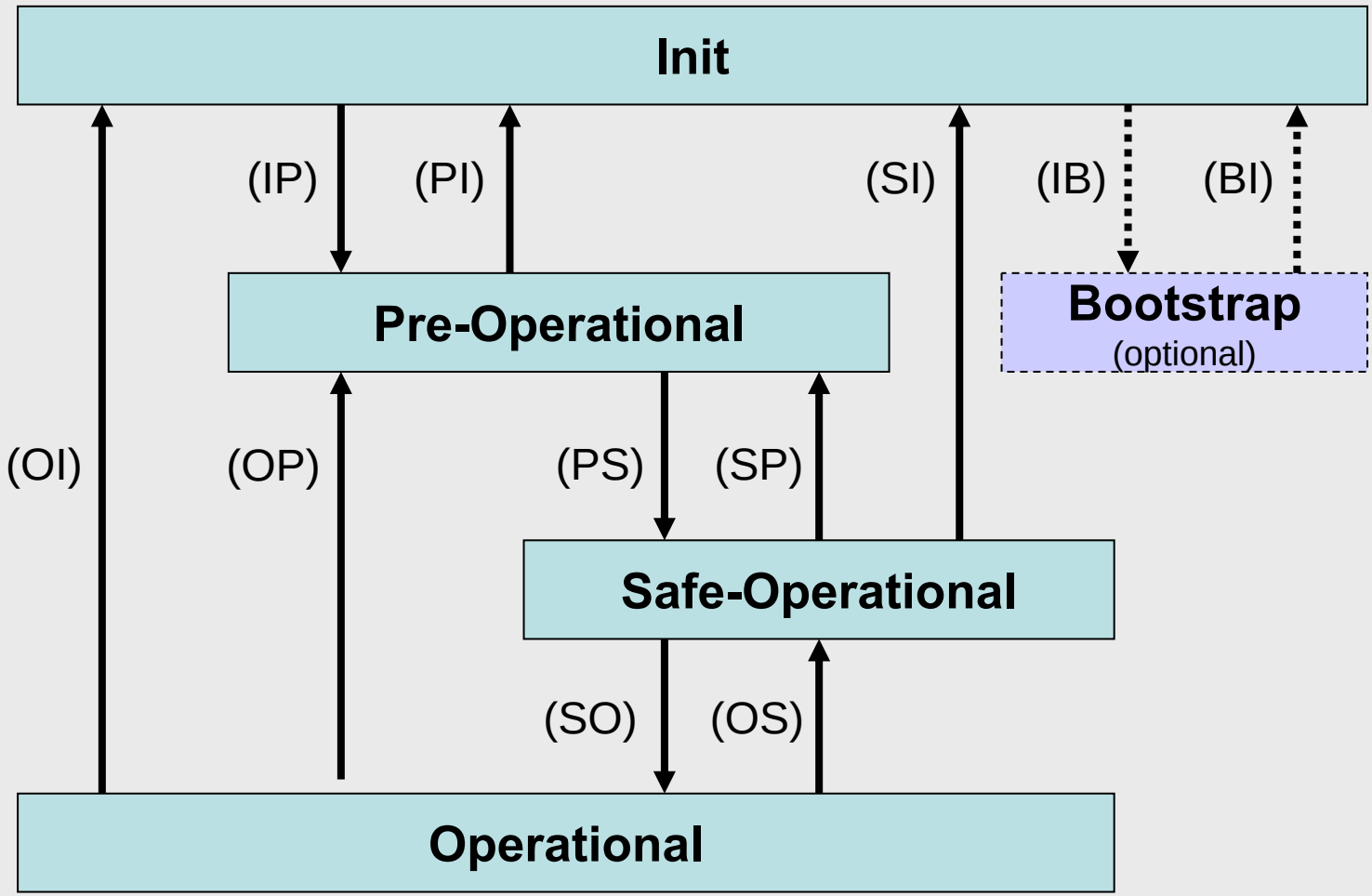
- EtherCAT State Machine
- Mailbox Interfaces and Protocols
  - Ethernet over EtherCAT
  - CANopen over EtherCAT
  - Filetransfer over EtherCAT
  - Servo Drive over EtherCAT
- Slave Information Interface (SII)


Application
....
Appli-
cation
HTTP,
FTP, ..
Application
e.g. CiA402 Drive Profile
RD / WR
...
File
Access
TCP, UDP
IP
Object Dictionary
PDO Mapping
SDO
PDO
VoE
FoE
EoE
CoE
CoE
Mailbox DLL
Application
Layer
(AL)
Process Data Interface (µC, SSI, I/O)
SII
EEPROM
Registers
Mailbox
Process Data
ESC Address Space (DPRAM)
0x0000
0x1000
SyncMan0 MbxOut
SyncMan1 MbxIn
SyncMan2
SyncMan3
FMMU n
FMMU n
EtherCAT Slave Controller (ESC)
EtherCAT Processing Unit
and Auto-Forwarder with Loop Back
PHY
Management
MII / EBUS
Port 0
MII / EBUS
Port 3
MII / EBUS
Port 2
MII / EBUS
Port 1
Data
Link
Layer
(DL)
RJ45
Trafo
PHY
PHY
Trafo
RJ45
LVDS
Con
Physical
Layer
(PL)
FCS
More Datagrams..
WKC
Mailbox Data
FPWR
HDR
WKC
Process Data
LRW
HDR
ECAT HDR
Ethernet HDR

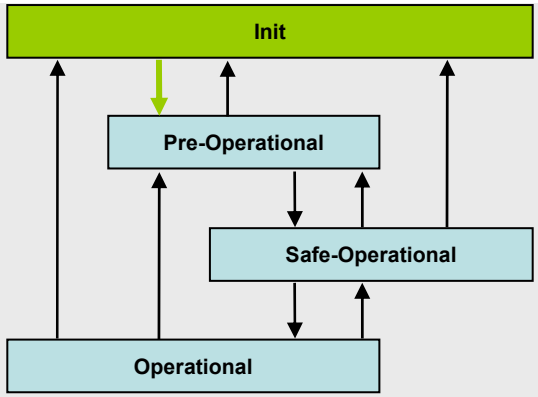
Purpose of EtherCAT State Machine

- State Machine is build upon the Data Link Layer
- Defines general communication states of EtherCAT slave devices
- Specifies the initialization and error handling of EtherCAT slave devices → Boot-up of the network
- States correspond to the communication relationship between master and slave
- Requested and current state of a slave device are reflected in the AL Control and AL Status registers
- Five states are defined:
  - ‘Init’, ‘Pre-Operational’, ‘Safe-Operational’, ‘Operational’
  - ‘Bootstrap’ optional state for firmware updates

EtherCAT State Machine



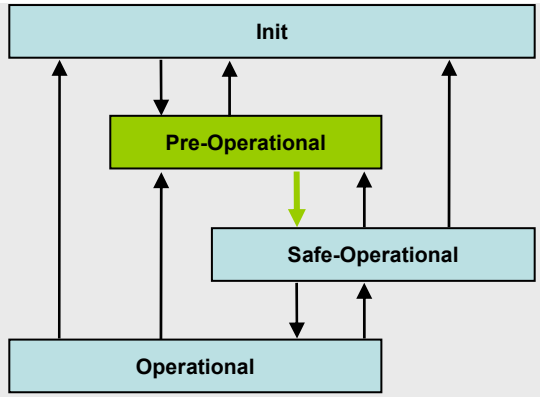
State Machine / Init



- **'Init' State**
- No communication on the Application Layer
- Master has access to the DL-Information registers

- **Transition to 'Pre-Operational'**
- Master configures register, at least:
  - DL Address register
  - Sync Manager channels for Mailbox communication
- Master requested 'Pre-Operational' state
  - sets AL Control register
  - wait for AL Status register confirmation

State Machine / Pre-Operational

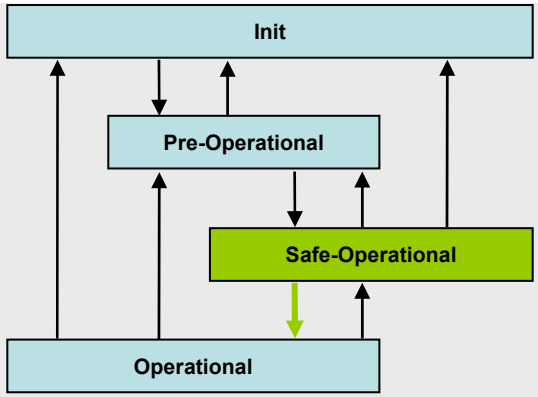


- **'Pre-Operational' State**
- Mailbox communication on the Application Layer
- No Process Data communication

- **Transition to 'Safe-Operational'**
- Master configures parameter using the Mailbox
  - e.g.: Process Data Mapping
- Master configures DL Register
  - SyncManager channels for Process Data communication
  - FMMU channels
- Master requested 'Safe-Operational' state

State Machine / Safe-Operational

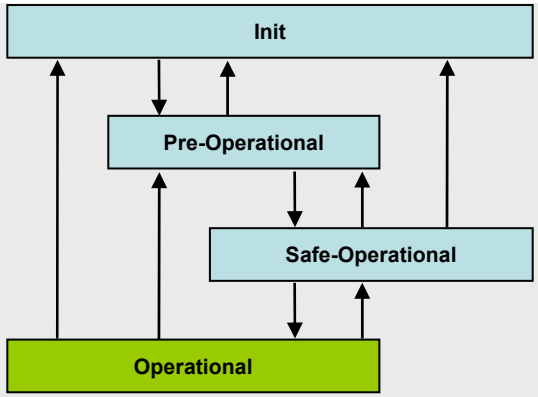
- **'Safe-Operational' State**
- Mailbox communication on the Application Layer
- Process Data communication, but only Inputs are evaluated – Outputs in 'Safe' state



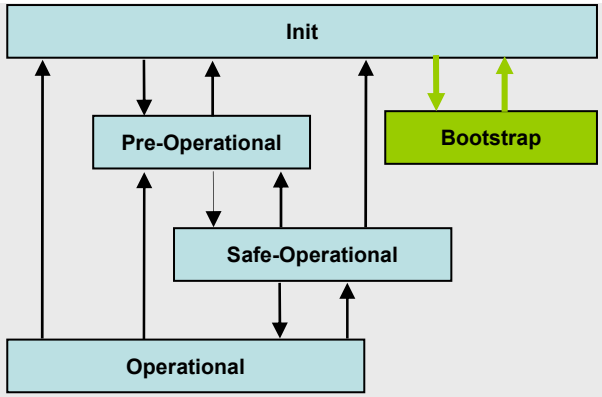
- **Transition to 'Operational'**
- Master sends valid Outputs
- Master requested 'Operational' state (AL Control/Status)

State Machine / Operational

- **‘Operational’ State**
- Inputs and Outputs are valid



State Machine / Bootstrap



- **'Bootstrap' State**
- 'Bootstrap' State is optional – but recommended if firmware updates necessary
- State changes only from and to 'Init'
- No Process Data communication
- Communication via Mailbox on Application Layer
- Special mailbox configuration possible, e.g. larger mailbox size
- Only FoE protocol available (possibly limited "file" range)

State Machine / Control and Status

- Requested and current state of a slave device are reflected in the AL Control and AL Status registers
  - AL Control (0x0120)
    Initiate State Transition of Device State Machine
  - AL Status (0x0130)
    Actual State of Device State Machine
  - AL Status Code (0x0134)
    Reason of error or other status code

Diagnosis at Application Layer

- AL Status Code (0x0134)
  - Error Codes (extract)

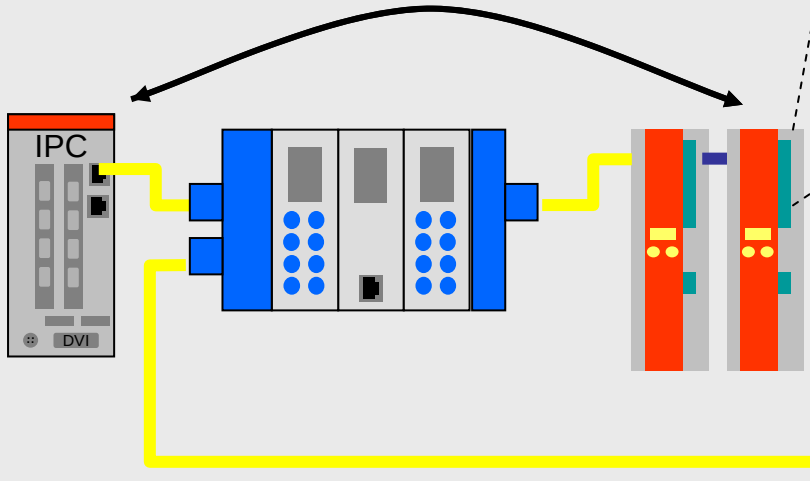
Code	Description
0x0000	No Error
0x0011	Invalid requested state change
0x0015	Invalid mailbox configuration
0x0018	No valid inputs available
0x0019	No valid outputs
0x001A	Synchronization error
0x001B	Sync manager watchdog

  - Further Status Codes (extract)

Code	Description
0x0021	Slave needs INIT
0x0022	Slave need PREOP
0x0030	Invalid DC Sync Configuration
0x0031	Invalid DC Latch Configuration

Mailbox Transfer

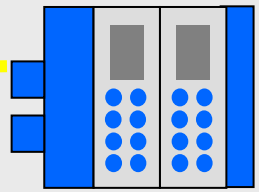
Mailbox Transfer for Parameter Data
Full duplex capable



Device Parameter (example)

Index	Value
0x1600	RxPDO Mapping
0x1A00	TxPDO Mapping
0x2000	Current control
0x2010	Velocity control
0x2040	Motor parameter
0x2070	Actual values
0x6040	Controlword
0x6041	Statusword

State "PreOperational"



Simple IO-Device
No Parameter
→ No Mailbox necessary

Purpose of Mailbox Transfer

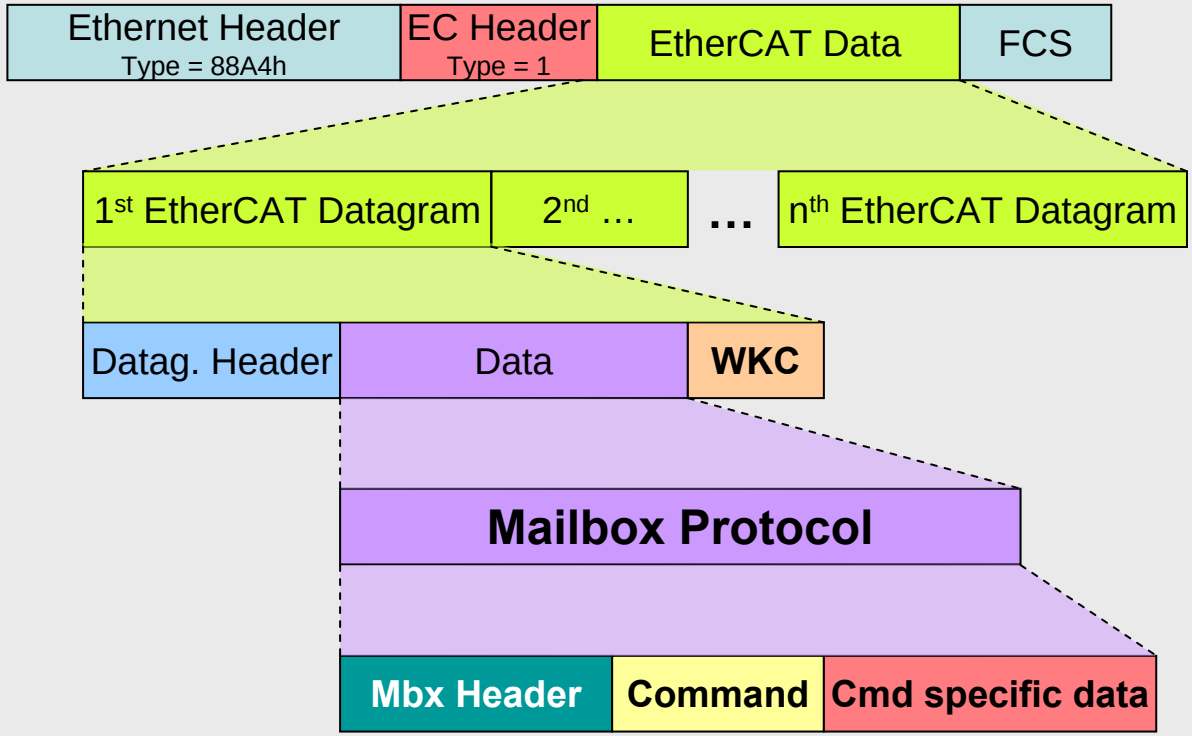
- Standard way to exchange Parameter Data
- The Mailbox Interface is optional – but recommended
- Needed if Process Data configurable or any other non cyclic services
- Full duplex capable
  (Slave can initiate a communication)
- 2 Sync Manager channels reserved
  - Sync Manager 0 : Master to Slave
  - Sync Manager 1 : Slave to Master
- Available at early stage of communication
  (State Pre-Operational)
- Multi protocol capable

Mailbox Protocol Types

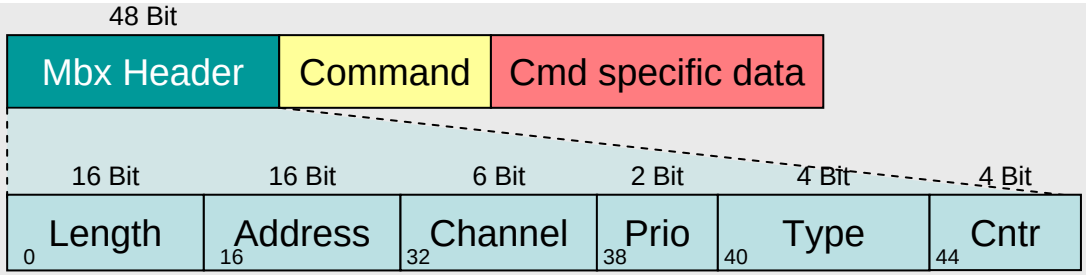
- Ethernet over EtherCAT (EoE)
  - Tunnels standard Ethernet Frames over EtherCAT
- CANopen over EtherCAT (CoE)
  - Access of a CANopen object dictionary and its objects
  - CANopen Emergency and optional event driven PDO messages
- File Access over EtherCAT (FoE)
  - Download and upload firmware and other ‘files’
- Servo Drive over EtherCAT (SoE)
  - Access the Servo Profile Identifier (IDN)
- Vendor specific Profile over EtherCAT (VoE)
  - First DWORD contains the Vendor ID, the next WORD contains a Vendor Type, the rest is vendor specific

Mailbox Interface

- Datagram within an EtherCAT Frame
- 2 Sync Manager channels (1 buffer-mode) reserved
  - Sync Manager 0 : Master to Slave Mbx
  - Sync Manager 1 : Slave to Master Mbx



Mailbox Header



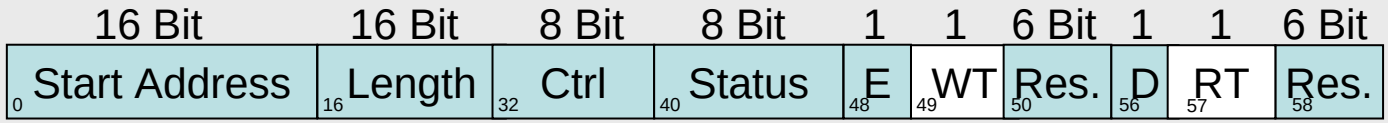
Length	Length of following data
Address	Station Address of originator
Channel	reserved for future use
Priority	reserved for future use
Type	Mailbox Type, Protocol identifier for following data 0 Mailbox Error 2 EoE (Ethernet over EtherCAT) 3 CoE (CANopen over EtherCAT) 4 FoE (File Access over EtherCAT) 5 SoE (Servo Drive over EtherCAT) 15 VoE (Vendor specific profile over EtherCAT)
Counter	Sequence number for duplicate detection Increments with every new mailbox service (only the values 1-7 will be used to be compatible with older versions).

Resilient Mailbox Procedure

- Reliable way of mailbox data exchange
- Mailbox control procedure
- Recover from lost frames
- No additional frames if no error
- Additional receive buffer required
- Extra counter in mailbox header needed
- HW/SW solution
  - SyncManager configuration register with toggle flags
  - SW-Mailbox-DL Layer for checking toggle bits

Mailbox Error Handling – Sync Manager

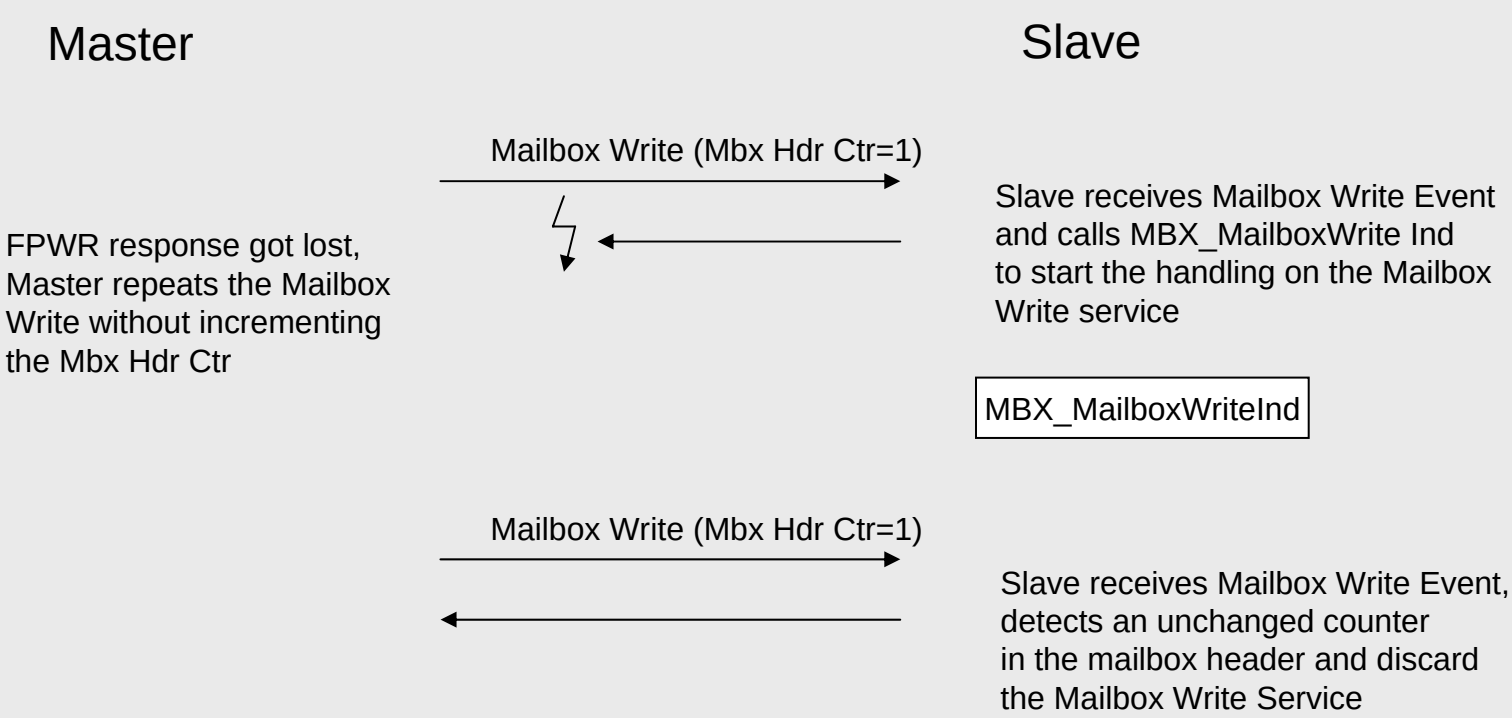
- Sync Manager channel configuration registers



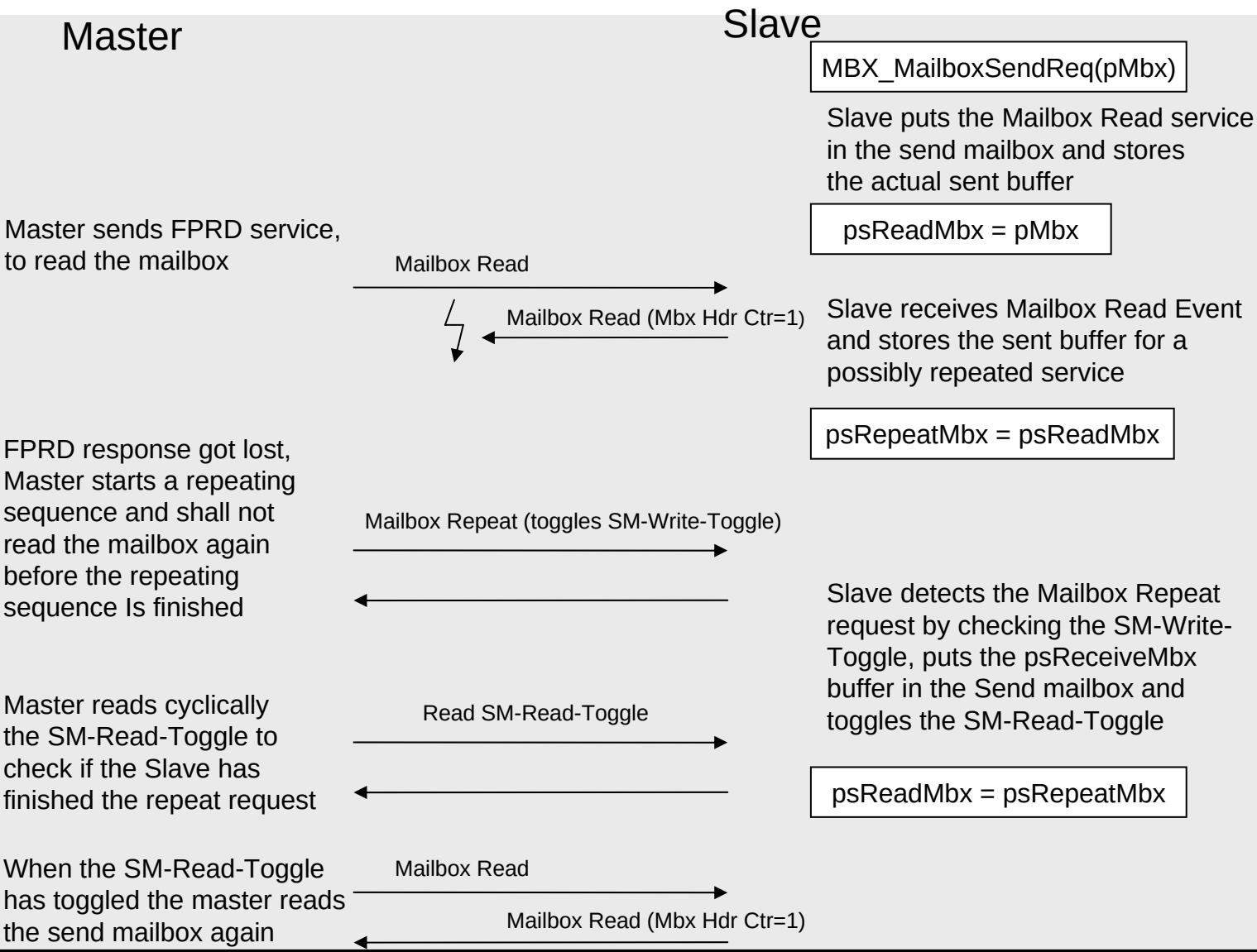
WT – SM Write Toggle
RT – SM Read Toggle

- Following:
  - Mailbox Error Handling - Write Example
  - Mailbox Error Handling - Read Example

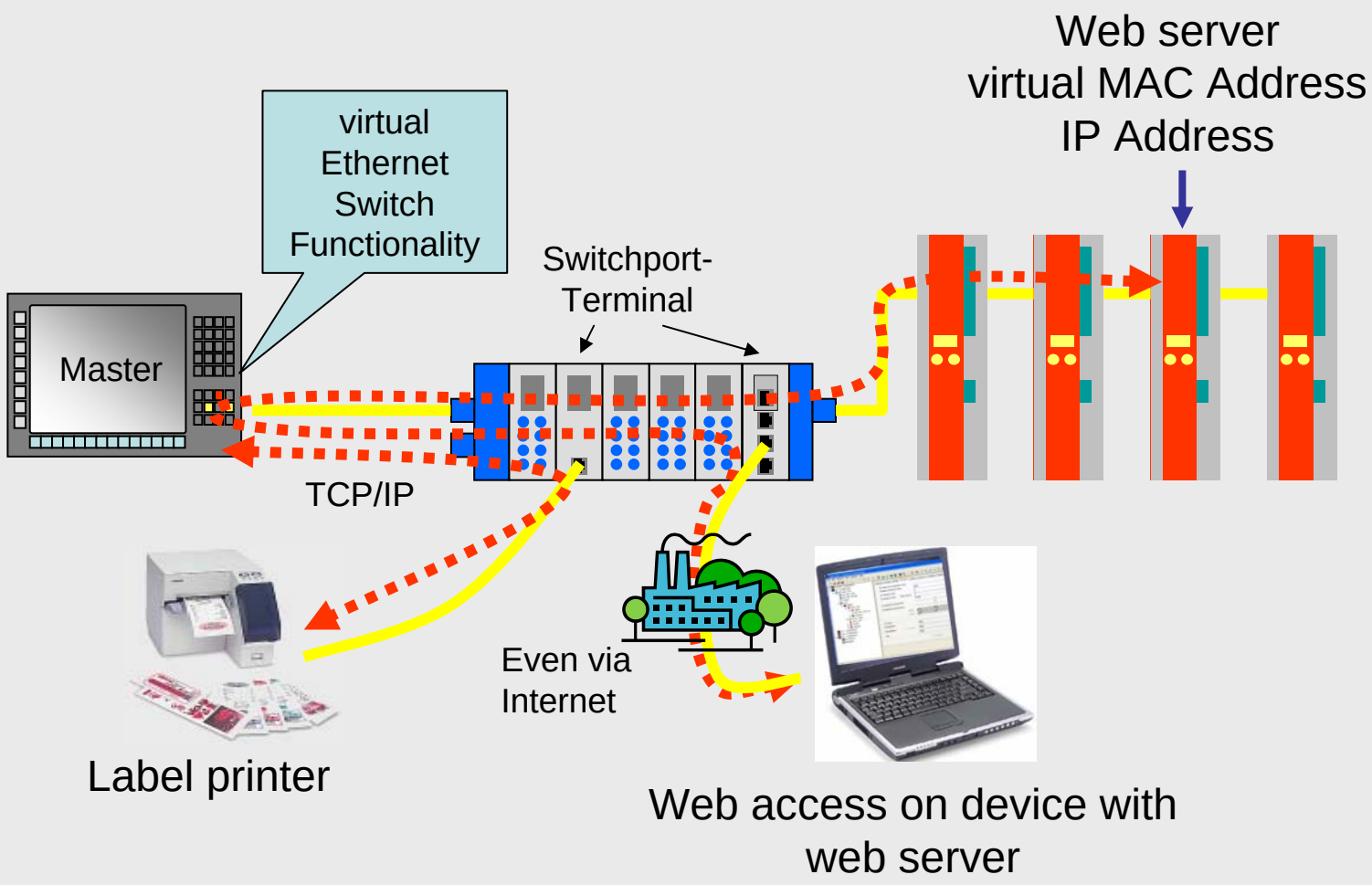
Mailbox Error Handling – Mailbox Write



Mailbox Error Handling – Mailbox Read



Ethernet over EtherCAT (EoE)

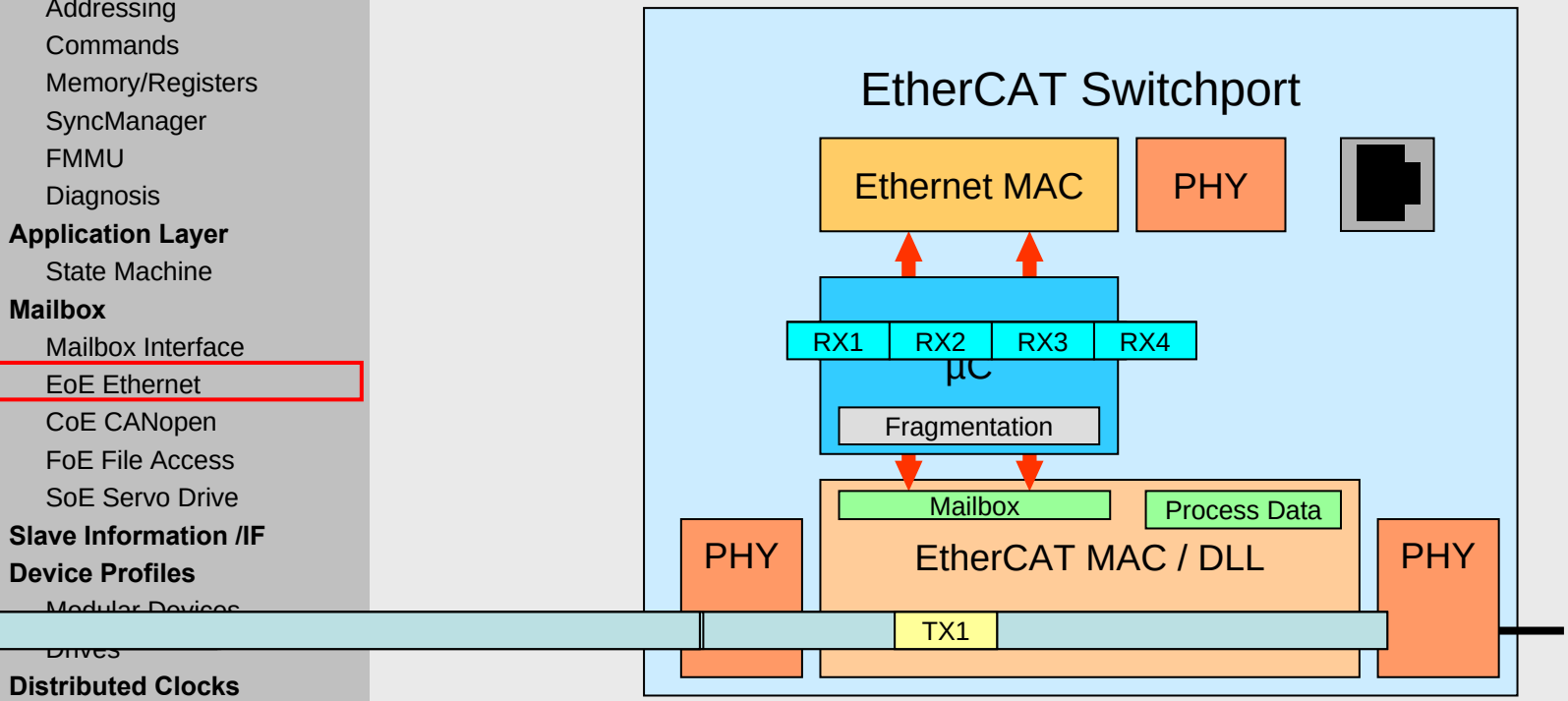


Purpose of Ethernet over EtherCAT (EoE)

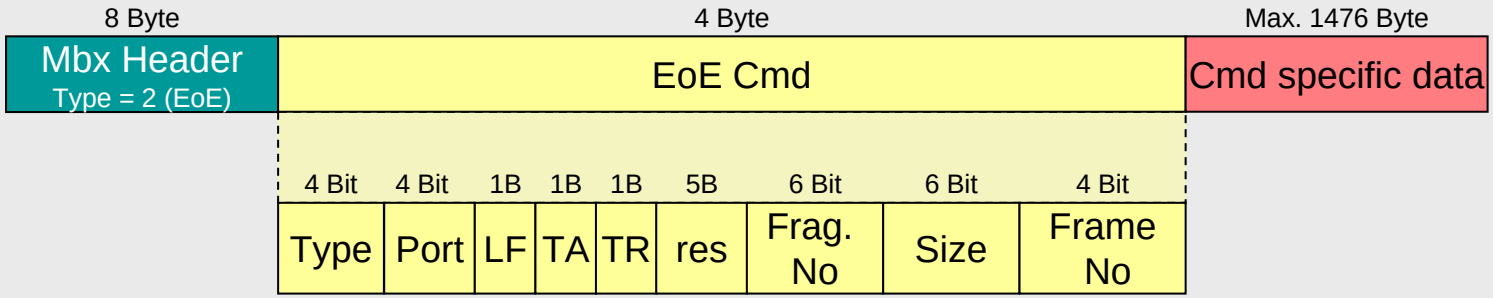
- Tunnels transparently Ethernet Frames over EtherCAT
- Tunneling reduces the cycle times without restrictions and to optimized available bandwidth
- Used for devices with TCP/IP stacks (e.g. Web Server) and for infrastructure devices like Switch Terminals
- Allows to access corresponding devices in the normal IP network in combination with a ‘Virtual Ethernet Switch’ (Layer 2) on the master side

EoE – Switchport: Any Ethernet Protocol

- Interface to any Ethernet Device or Network
- Ethernet Frames are inserted in EtherCAT Protocol



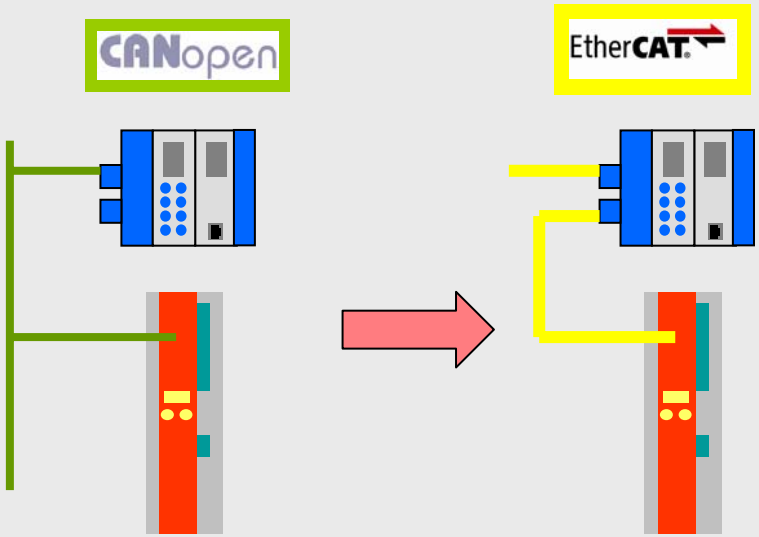
EoE – Frame Header



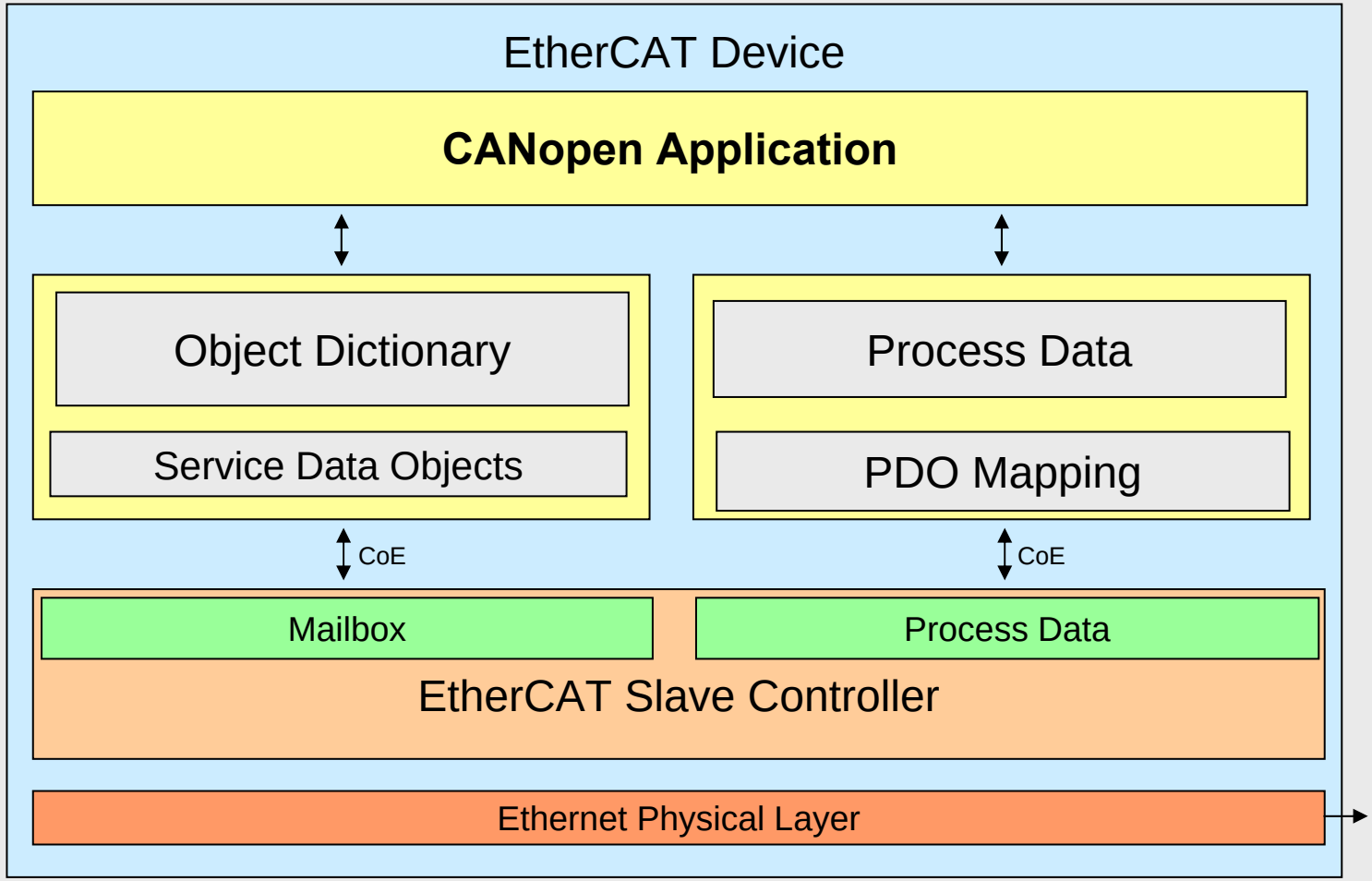
Type	EoE Frame Type 0x00 EoE Fragment Request 0x01 Initiate EoE Request 0x02 IP Parameter Request 0x03 IP Parameter Response 0x04 Set MAC Address Filter Request 0x05 Set MAC Address Filter Response
Port	Selected Port
LF	Last fragment
TA	Time stamp appended (only if LF=1)
TR	Time stamp request
Fragment No	Fragment Number of the Ethernet Frame fragment
Size	Complete size of Ethernet Frame
Frame No	Number of the Ethernet Frame

Purpose of CANopen over EtherCAT (CoE)

- Recommended protocol for Service Data Access
  - Configuration of communication parameter
  - Configuration of device specific parameter
- Easy migration path from CANopen Devices to EtherCAT CoE Devices
  - Protocol Stacks can be re-used



CANopen over EtherCAT Device Architecture



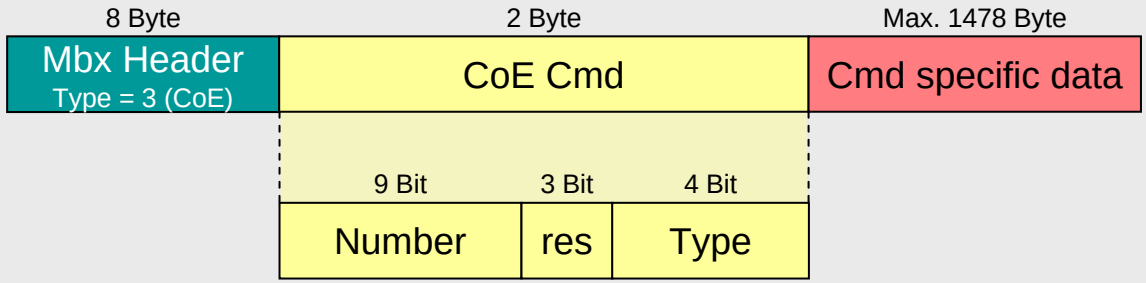
CANopen over EtherCAT (CoE)

- SDO: Access to a CANopen object dictionary ✓
  - Download and Upload of parameters
  - Standard Process Data Mapping (PDO Mapping)
  - Full Access to CANopen Profiles
- PDO: Process Data Objects ✓
  - Direct PDO transfer
  - Remote Transmission Requests of PDOs
- Emergency Messages ✓
- Network Management ✓

Standard CANopen Features

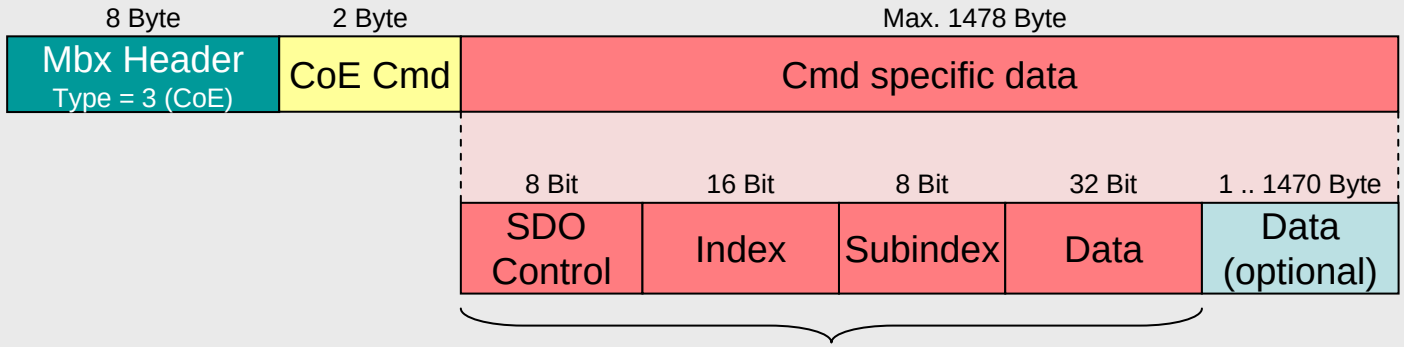
- + Object Dictionary Information (SDO Information)
  - Upload of object dictionary (identifier lists)
  - Upload of object description
  - Upload of entry descriptions

CoE –Frame Header



Number	PDO Number (PDO transfer only)
Type	Message Type 0 reserverd 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 reserved for future use

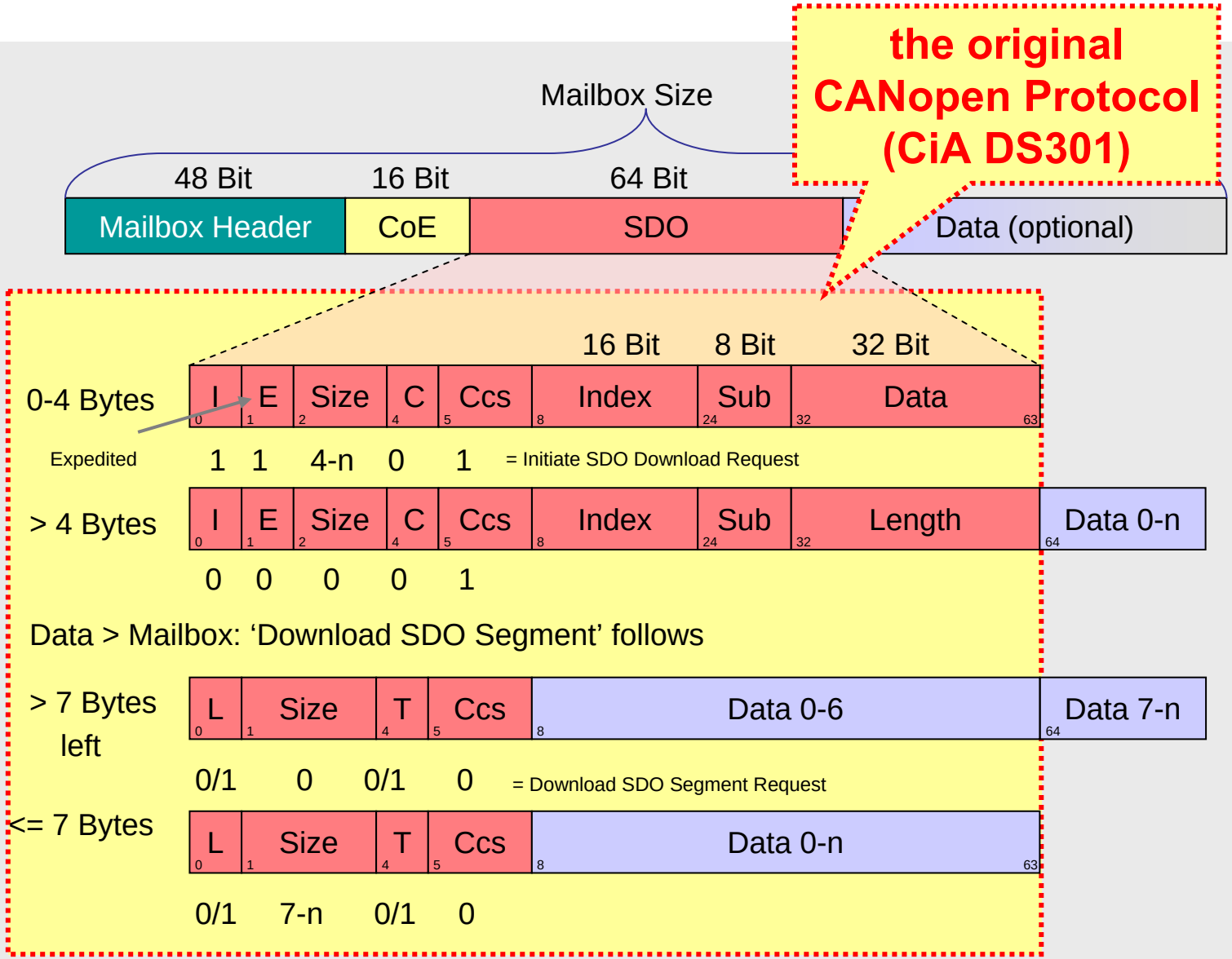
CoE – Standard CANopen Frames



Standard CANopen Frame

SDO Control	**Standard CANopen SDO Services**
Index	**Object Addressing by Index …**
Subindex	**… and Subindex**
Data	**Data for the SDO-Service**
Data (optional)	**Optionally more than 4 bytes of data can be sent with one frame. Full mailbox size usable**

Example: 'Initiate SDO Download' Request



Optional SDO Extensions to CANopen DS 301

- Breaking of the 8 byte border
  - Full mailbox size usable
    → Block transfer unnecessary
  - 'Initiate SDO Download' request /
    'SDO Upload' response can contain data after SDO header
  - 'Download SDO Segment' request /
    'Upload SDO Segment' response can contain more than 7 bytes of data

- Downloading and Uploading all Subindices at once
  - Bit 4 of the Initiate 'SDO Download / Upload' request header indicates a 'Complete Access' to an Index
  - Sub Index field contains the start Subindex
    - 0: Complete Index with all Subindices
    - 1: Complete Index without Subindex 0

CoE: Optional SDO Information Protocol

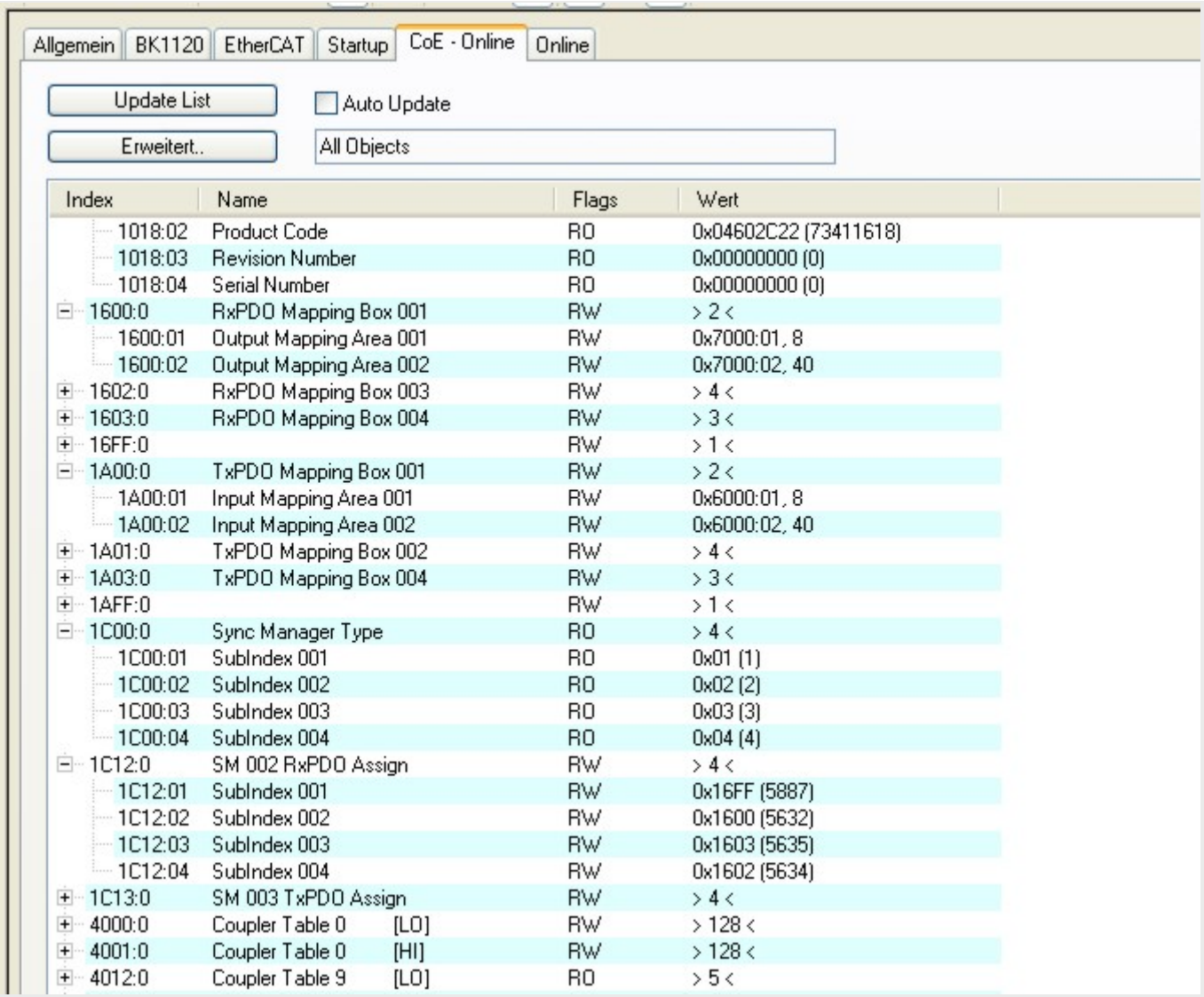
- Access to object lists
  - list of all supported object identifiers
  - list of all object identifiers that can be mapped in PDOs
  - list of all object identifiers that should be included in a backup
- Access to object descriptions
  - Descriptions of objects – as defined in DS 301
- Access to entry descriptions
  - Descriptions of object entries (sub index) – as defined in DS 301

CoE – Object Dictionary

- Devices with an EtherCAT-CoE and a CANopen interface are possible with the same object dictionary

Index Range	Meaning
0x0000 – 0x0FFF	Data Type Description
0x1000 – 0x1FFF	Communication objects • Device Type, Identity, PDO Mapping – like defined in DS 301 • Objects defined in DS 301 not needed are reserved for EtherCAT • Additional objects (Sync Manager Communication Type, Sync Manager PDO Assignment) located in unused areas of DS 301
0x2000 – 0x5FFF	Manufacturer specific
0x6000 – 0x9FFF	Profile specific
0xA000 – 0xFFFF	reserved

CoE – Online Object Dictionary



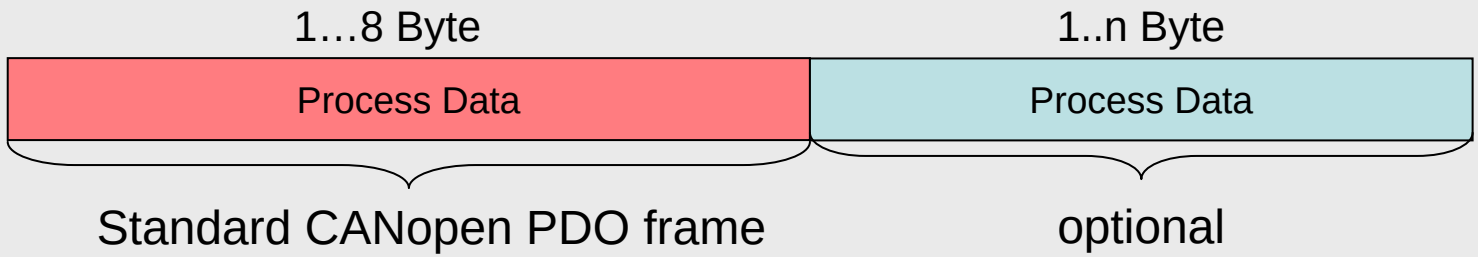
Index	Name	Flags	Wert
1018:02	Product Code	RO	0x04602C22 (73411618)
1018:03	Revision Number	RO	0x00000000 (0)
1018:04	Serial Number	RO	0x00000000 (0)
1600:0	RxPDO Mapping Box 001	RW	> 2 <
1600:01	Output Mapping Area 001	RW	0x7000:01, 8
1600:02	Output Mapping Area 002	RW	0x7000:02, 40
1602:0	RxPDO Mapping Box 003	RW	> 4 <
1603:0	RxPDO Mapping Box 004	RW	> 3 <
16FF:0		RW	> 1 <
1A00:0	TxPDO Mapping Box 001	RW	> 2 <
1A00:01	Input Mapping Area 001	RW	0x6000:01, 8
1A00:02	Input Mapping Area 002	RW	0x6000:02, 40
1A01:0	TxPDO Mapping Box 002	RW	> 4 <
1A03:0	TxPDO Mapping Box 004	RW	> 3 <
1AFF:0		RW	> 1 <
1C00:0	Sync Manager Type	RO	> 4 <
1C00:01	SubIndex 001	RO	0x01 (1)
1C00:02	SubIndex 002	RO	0x02 (2)
1C00:03	SubIndex 003	RO	0x03 (3)
1C00:04	SubIndex 004	RO	0x04 (4)
1C12:0	SM 002 RxPDO Assign	RW	> 4 <
1C12:01	SubIndex 001	RW	0x16FF (5887)
1C12:02	SubIndex 002	RW	0x1600 (5632)
1C12:03	SubIndex 003	RW	0x1603 (5635)
1C12:04	SubIndex 004	RW	0x1602 (5634)
1C13:0	SM 003 TxPDO Assign	RW	> 4 <
4000:0	Coupler Table 0 [LO]	RW	> 128 <
4001:0	Coupler Table 0 [HI]	RW	> 128 <
4012:0	Coupler Table 9 [LO]	RO	> 5 <

CoE – New objects for EtherCAT

Index	Meaning
0x1C00	SyncManager Communication Type
0x1C10 – 0x1C2F	SyncManager PDO Assign
0x1C30 – 0x1C4F	SyncManager Parameter

- SyncManager Communication Type
  - Subindex (1-32) defines communication type of the corresponding Sync Manager channel
    - Mailbox Out (= 1 buffer write)
    - Mailbox In (= 1 buffer read)
    - Process Data Out (= 3 buffer write)
    - Process Data In (= 3 buffer read)
- SyncManager PDO Assign
  - Contains a list of assigned PDOs for each Sync Manager channel (Index of PDO mapping objects)
  - Assigned PDO in Subindex 1 to n describe the process data parts of the Sync Manager channel
- SyncManager Parameter
  - SubIdx 1: Synchronization type (Freerun, synchron, DC Sync0, DC Sync1, SyncSm0 .. SyncSm1F)
  - SubIdx 2: Cycle time
  - SubIdx 3: Shift time

CoE – Process Data Objects



- Standard CANopen PDO frames can be used
- CANopen PDO mapping used for EtherCAT too
- PDOs usually are related to their „own“ Sync Manager channels
  - Cyclic data transmission
  - No PDO communication parameters necessary
- PDOs can be transmitted by the mailbox for acyclic communication

CoE – PDO Process Data mapping

- No mapping protocol implemented for very simple devices
  - Fixed process data
  - Readable via EEPROM – no SDO protocol necessary
- Readable PDO Mapping
  - Fixed process data mapping
  - Readable via SDO
- Selectable PDO Mapping
  - Multiple fixed PDO – selectable via CoE object (1C1xh)
  - Selectable via SDO (required)
- Variable PDO Mapping
  - Configurable via CoE or SoE required
  - Writable PDO content

CoE – PDO Configuration

General | EtherCAT | Process Data | Mailbox | DC | Startup | CoE - Online | Online

Sync Manager:

SM	Size	Type	Flags
0	246	MbxOut	
1	246	MbxIn	
2	3	Outputs	
3	5	Inputs	

PDO List:

Index	Size	SM	Name	Flags
0x1600	3.0	2	RxPDO 001 mapping	F
0x1610	4.0		RxPDO 017 mapping	F
0x1A00	5.0	3	TxPDO 001 mapping	F
0x1A10	6.0		TxPDO 017 mapping	F

PDO Assignment (0x1C12):

- [x] 0x1600
- [ ] 0x1610 (excluded by 0x1600)

PDO Content (0x1600):

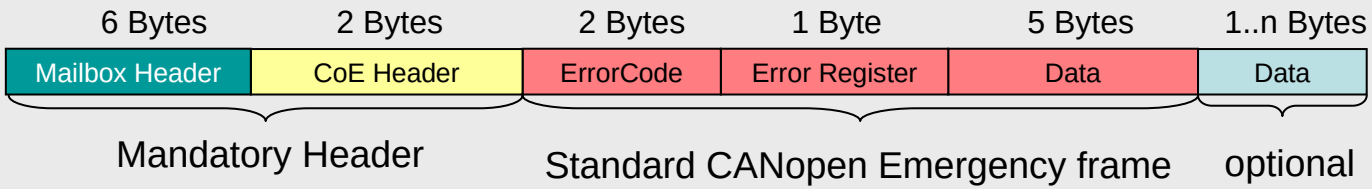
Index	Size	Offs	Name	Type
0x3001:1	1.0	0.0	Control	USINT
0x3001:2	2.0	1.0	Data	UINT
		3.0		

Download

- [x] PDO Assignment
- [x] PDO Configuration

Load PDO info from device

CoE – Emergency Message



- Standard CANopen Emergency frames can be used
- More than 5 bytes of data can be send optionally with one frame

File Access over EtherCAT (FoE)

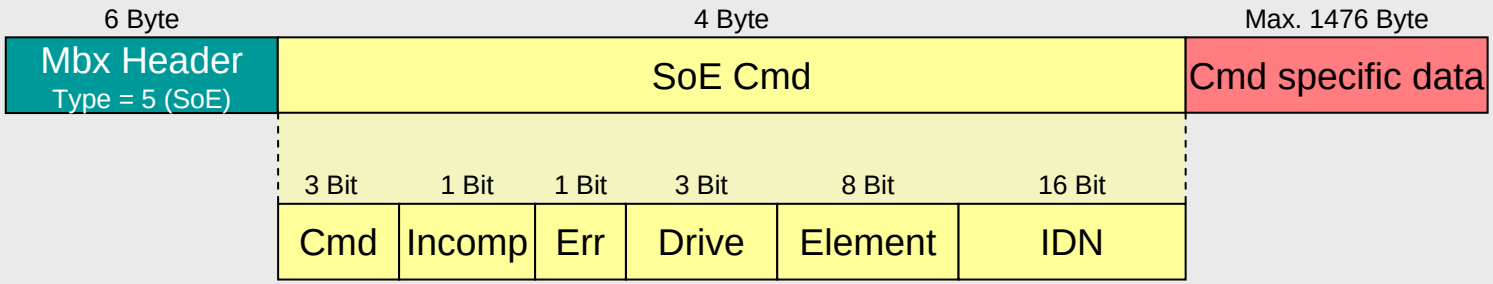
- Similar to TFTP (Trivial File Transfer Protocol, RFC 1350)
- Simple to implement – suitable for bootstrap loaders
- 6 Services are defined:
  - WRQ: Write request with “file name”
  - RRQ: Read request with “file name”
  - DATA: Data block (full mailbox size used)
  - ACK: Acknowledgment of DATA and WRQ requests
  - ERR: Error notification with predefined error codes
  - BUSY: Busy notification in case of longer procedures, extension to TFTP (e.g. erasing of flash modules)
- Special mailbox configuration for bootstrap mode possible
  - Fixed addresses and fixed size of the mailbox
  - Configuration defined by device (EEPROM)

Servo Drive over EtherCAT (SoE)

- Implements Service Channel
  - Read / Write to several elements of an IDN (Ident number)
  - Support of Procedure Commands
  - Slave Info

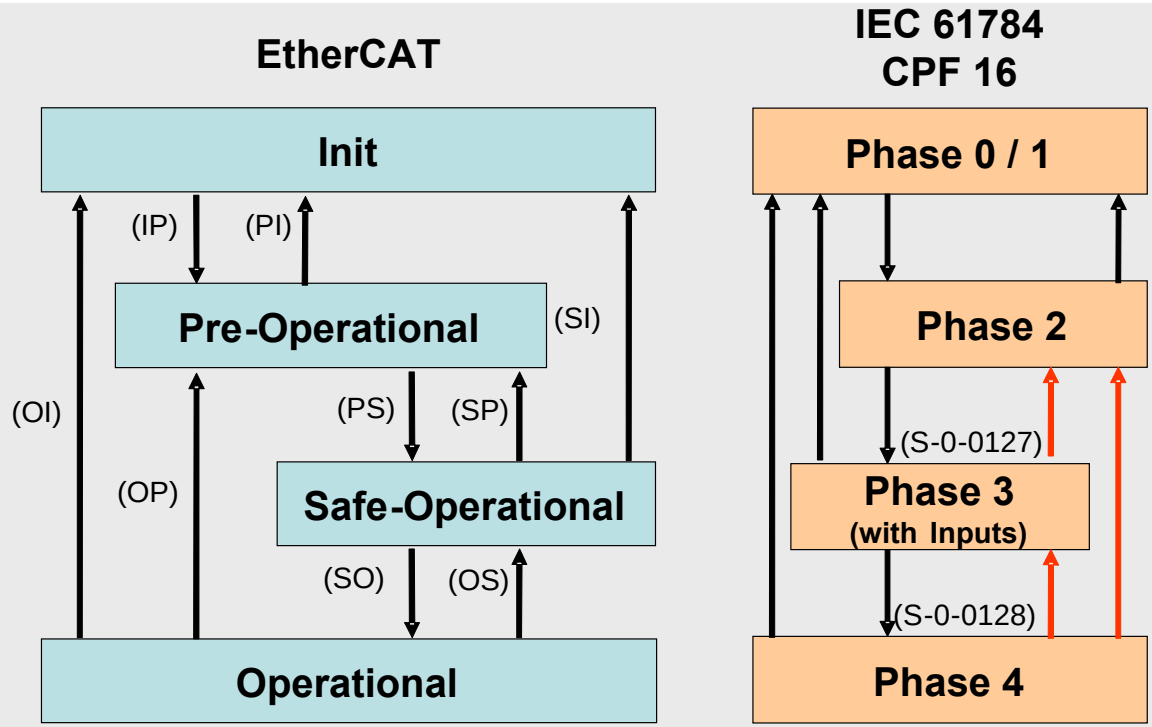
- The mapping of the IEC 61800-7-1 Annex D (SERCOS™) on EtherCAT is described in IEC 61800-7-3 Annex D

SoE – Frame Header

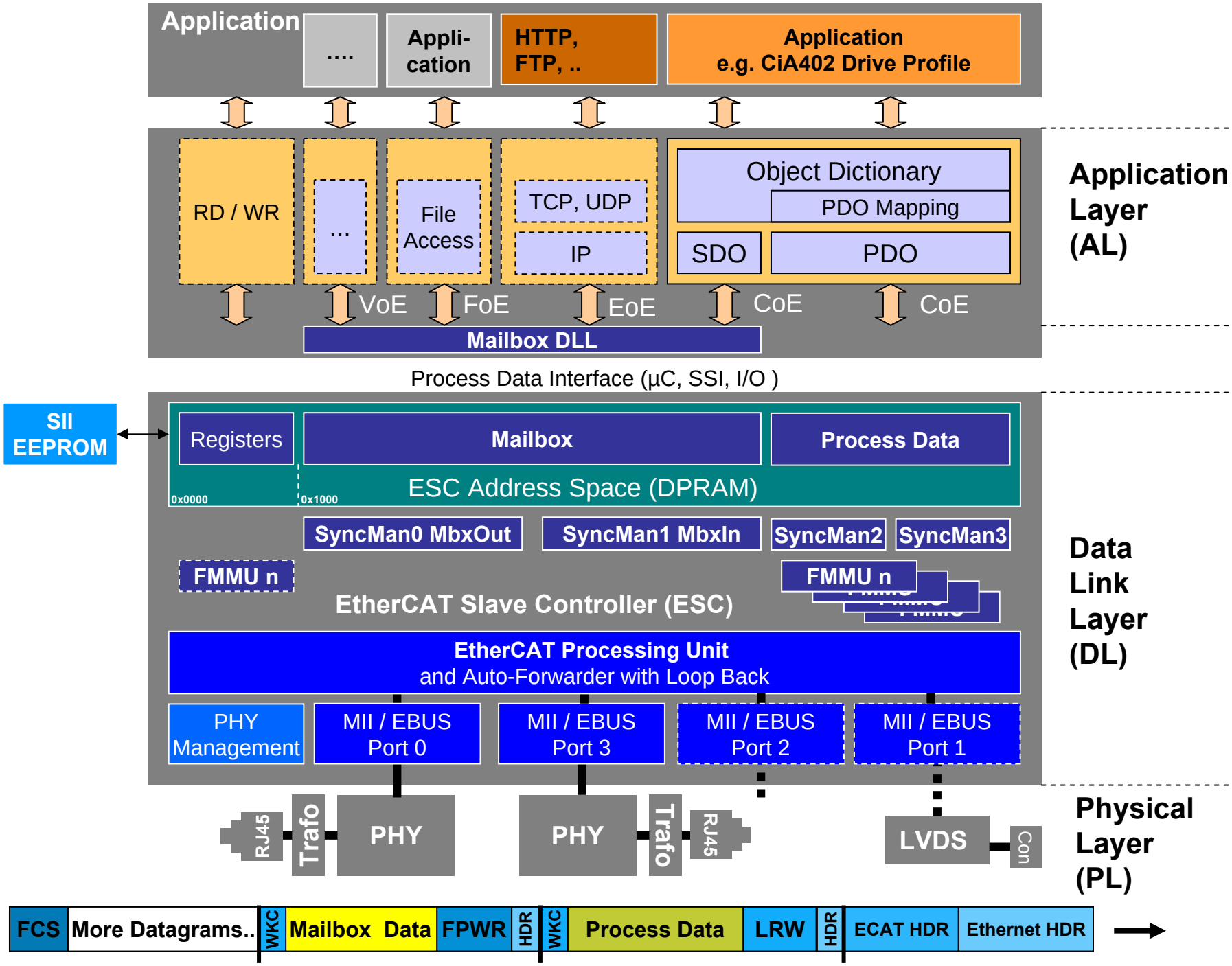


Command	**Command Type** **Read Request, Read Response, Write Request, Write Response, Notification**
Incomplete	**Indicates if execution of another service is needed to complete the operation**
Error	**Indicates if an Error has occurred**
Drive	**Contains the address of the drive inside the slave device that is addressed**
Element	**Contains the ElementFlags. There is a single Flag for each element of an IDN indicating which elements of the object addressed by the IDN are accessed**
IDN	**contains IDN according to IEC 61800-7-2 Annex D** **or an indicator for fragments left in case of segmented service**

SoE Communication phases



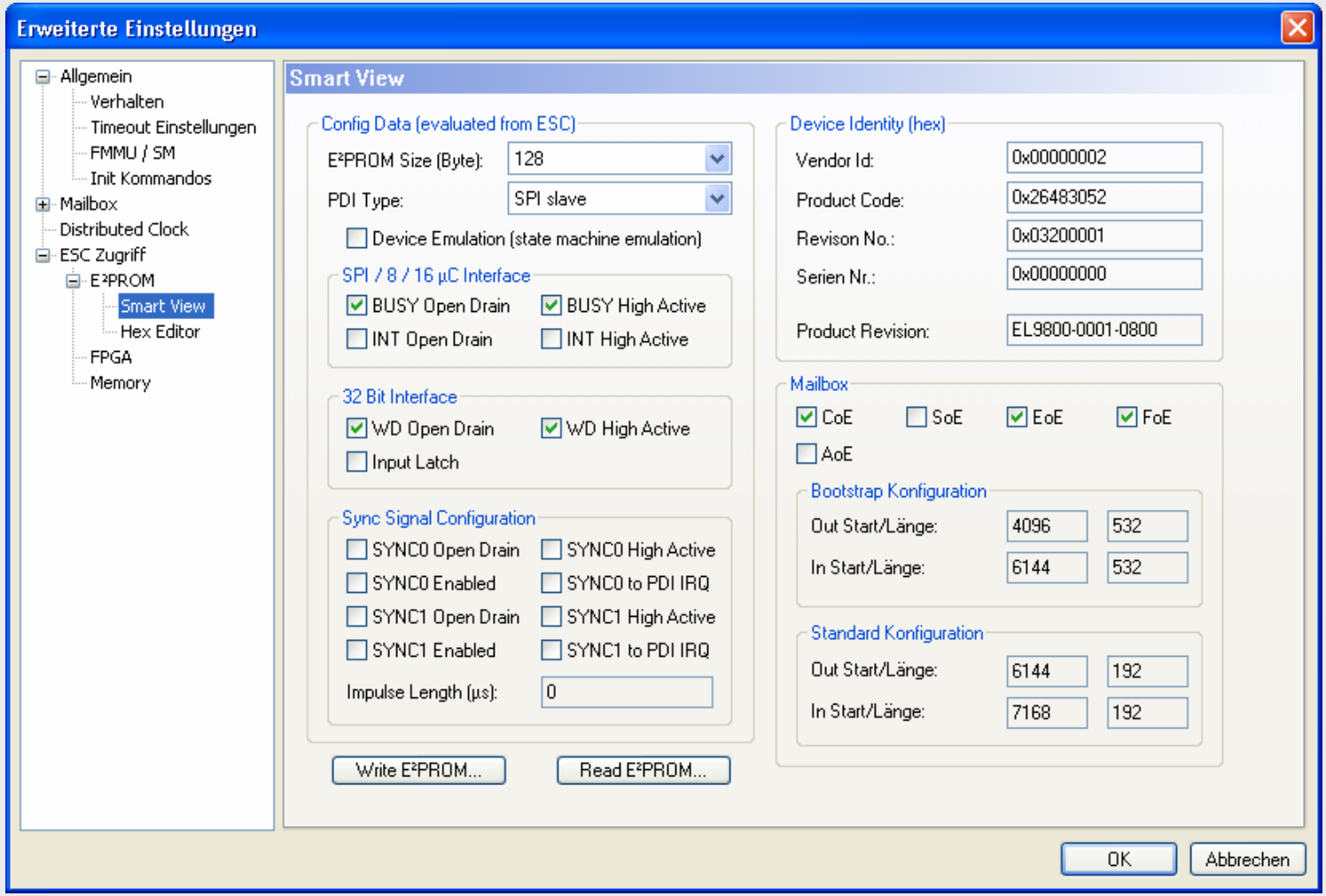
- SERCOS communication phases (CPs) comparable to EtherCAT state machine
- Phases 0 and 1 covered by the 'Init'
- Phase 2 corresponds to 'Pre-Operational'
  allows access to the IDNs via 'service channel' (SoE).
- Phase 3 mapped to 'Safe-Operational'
  slave shall transmit valid inputs, ignore outputs from the master.
- 'Operational' corresponds to phase 4
  all inputs and outputs are valid.


Application
....
Appli-
cation
HTTP,
FTP, ..
Application
e.g. CiA402 Drive Profile
RD / WR
...
File
Access
TCP, UDP
IP
Object Dictionary
PDO Mapping
SDO
PDO
VoE
FoE
EoE
CoE
CoE
Mailbox DLL
Application
Layer
(AL)
Process Data Interface (µC, SSI, I/O)
SII
EEPROM
Registers
Mailbox
Process Data
ESC Address Space (DPRAM)
0x0000
0x1000
SyncMan0 MbxOut
SyncMan1 MbxIn
SyncMan2
SyncMan3
FMMU n
FMMU n
EtherCAT Slave Controller (ESC)
EtherCAT Processing Unit
and Auto-Forwarder with Loop Back
PHY
Management
MII / EBUS
Port 0
MII / EBUS
Port 3
MII / EBUS
Port 2
MII / EBUS
Port 1
Data
Link
Layer
(DL)
RJ45
Trafo
PHY
PHY
Trafo
RJ45
LVDS
Con
Physical
Layer
(PL)
FCS
More Datagrams..
WKC
Mailbox Data
FPWR
HDR
WKC
Process Data
LRW
HDR
ECAT HDR
Ethernet HDR

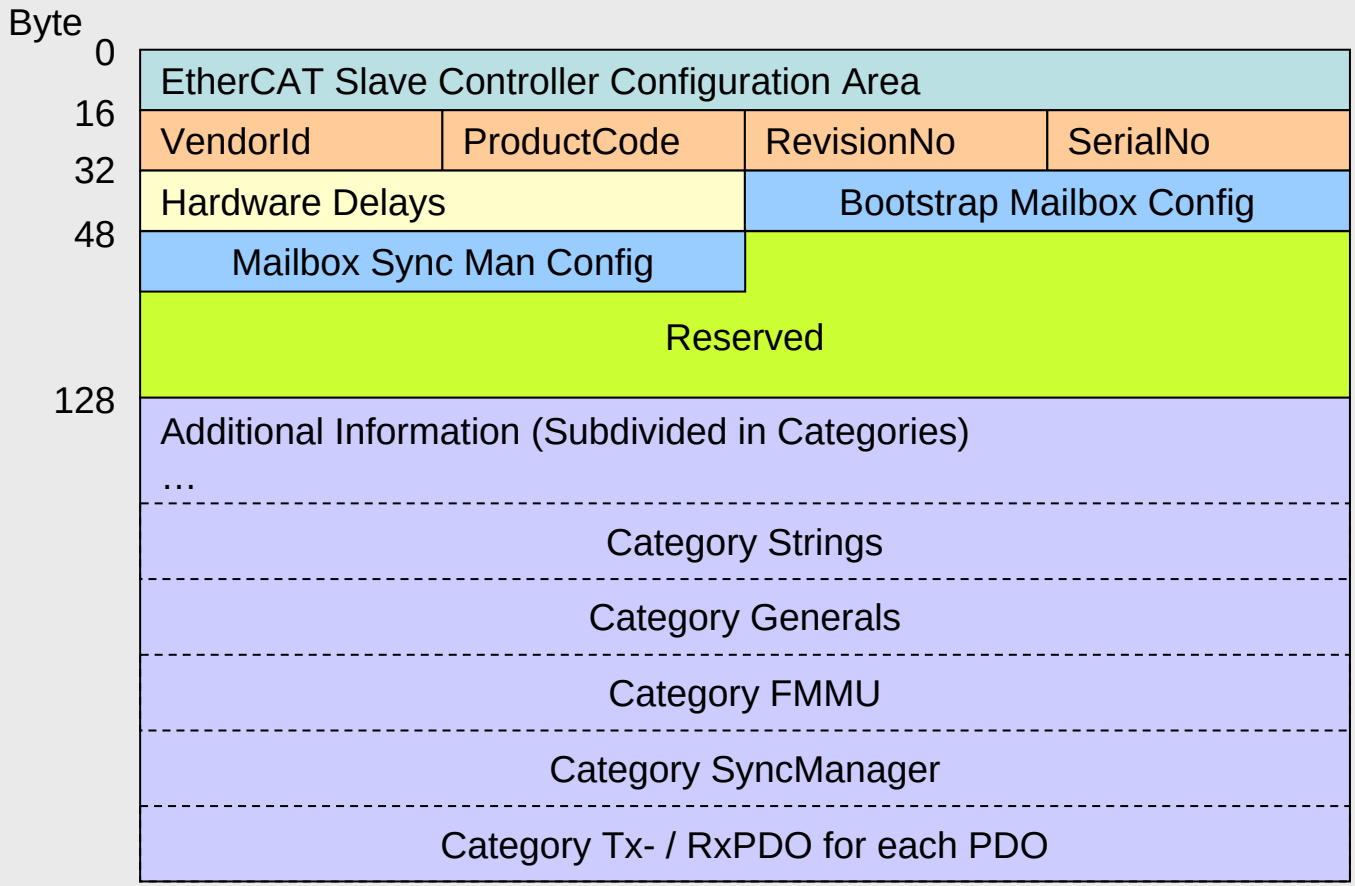
Purpose of Slave Information Interface

- The mandatory Slave Information Interface (SII) consists of all objects that can be stored persistently.
- The information is stored in an EEPROM
- EtherCAT Slave Controller has SPI interface to EEPROM
  - 1 kBit … 4 MBit
- The SII contains
  - boot configuration data
  - device identity (mandatory)
    - Vendor Id, Product Code, Revision No, Serial No
    - Same information in CoE object 0x1018
  - application information data
- Contains additional information (optional)
  - Subdivided in categories

SII – EEPROM content



SII – EEPROM Layout

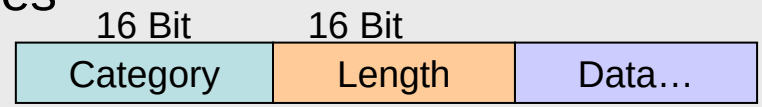


EEPROM – Slave Configuration Area

EEPROM Word Address	Parameter	Description	Corresp. Register
0	PDI Control	Initialization value for PDI Control register (EEPROM ADR 0x0000.9 is also mapped to register 0x0110.2)	0x0140 0x0141
1	PDI configuration,	Initialization value for PDI Configuration register Depends on the selected PDI, Configuration of Sync0 and Sync1 Pin	0x0150 0x0151
2	Pulse length of SYNC signals	Initialization value for Pulse Length of SYNC Signals register in Units of 10 ns	0x0982 0x0983
3	Extended PDI configuration	Initialization value for extended PDI Configuration register	0x0152 0x0153
4	Configured Station Alias	Initialization value for Configured Station Alias Address register	0x0012 0x0013
5, 6	Reserved		
7	Checksum	CRC of first 6 words (x^8+x^2+x+1, initial value 0xFF)	

SII – Categories

- Contains optional information
- Divided in categories
  - Standard category(s)
  - Vendor category(s)
- Same header for all categories
  - Category Type
  - Word-Length of data



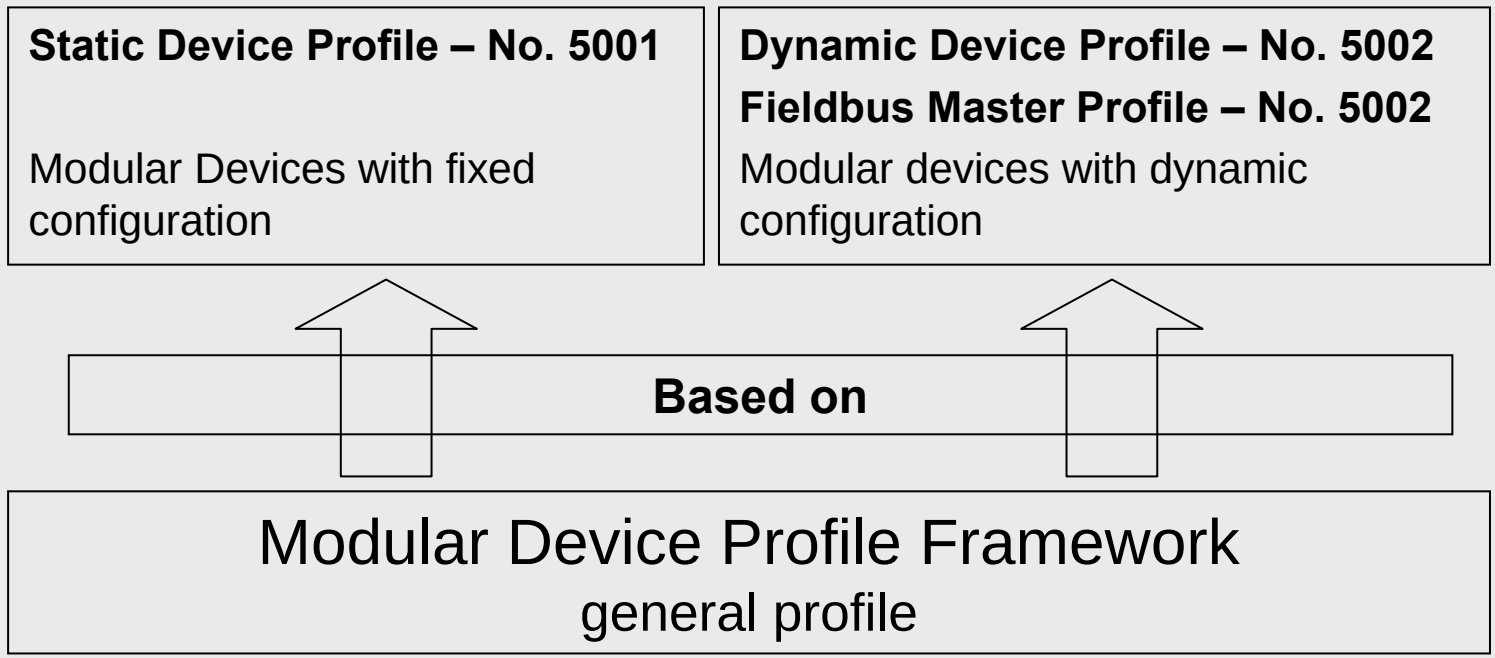
Category	Meaning
STRINGS	Text strings
General	Device Information
FMMU	FMMU usage
SyncManger	Modes of operation, Enable
TxPDO	Transmit PDO Entries
RxPDO	Receive PDO Entries

Device Profiles – Motivation

- EtherCAT supports complex slaves
- Slaves with multiple channels are common
- Channels consists of
  - Process data (Inputs/Outputs)
  - Configuration
  - Diagnosis
  - …
- Different channels may have different functionality

→ Profile for Modular Devices

Device Profiles - Overview



Profiles are EtherCAT specific!

Structuring of Object Dictionary

- Mapping of PDO like simple devices

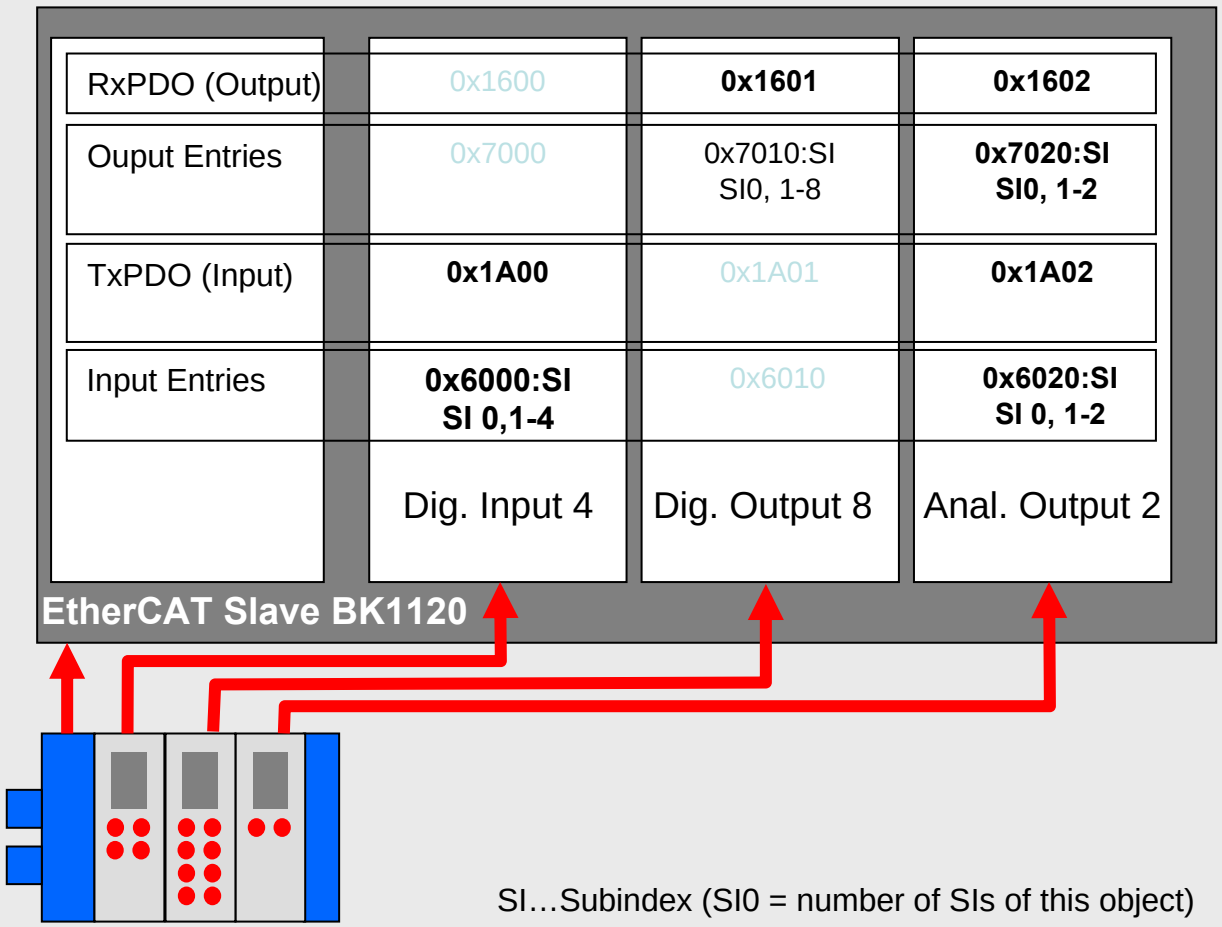
- Basic Idea of Mapping SDO is to have
  - Several areas with dedicated functions
  - Each IO Channel can have several objects in this areas
  - The number of objects per IO Channel depends upon implementation
  - Up to 4096 Channels can be managed by a single slave

Object Dictionary of Dynamic Device Profile

- 0x0000 – 0x0FFF Data Type Area
- 0x1000 – 0x1FFF: Communication Area
- 0x2000 – 0x5FFF: Manufacturer specific Area
- **0x6000 – 0x6FFF: Input Area**
- **0x7000 – 0x7FFF: Output Area**
- **0x8000 – 0x8FFF: Configuration Area**
- **0x9000 – 0x9FFF: Information Area**
- **0xA000 – 0xAFFF: Diagnosis Area**
- **0xB000 – 0xBFFF: Service Transfer Area**
- 0xC000 – 0xEFFF: Reserved Area
- **0xF000h – 0xFFFF: Device Area**

Example for Dynamic Device Profile

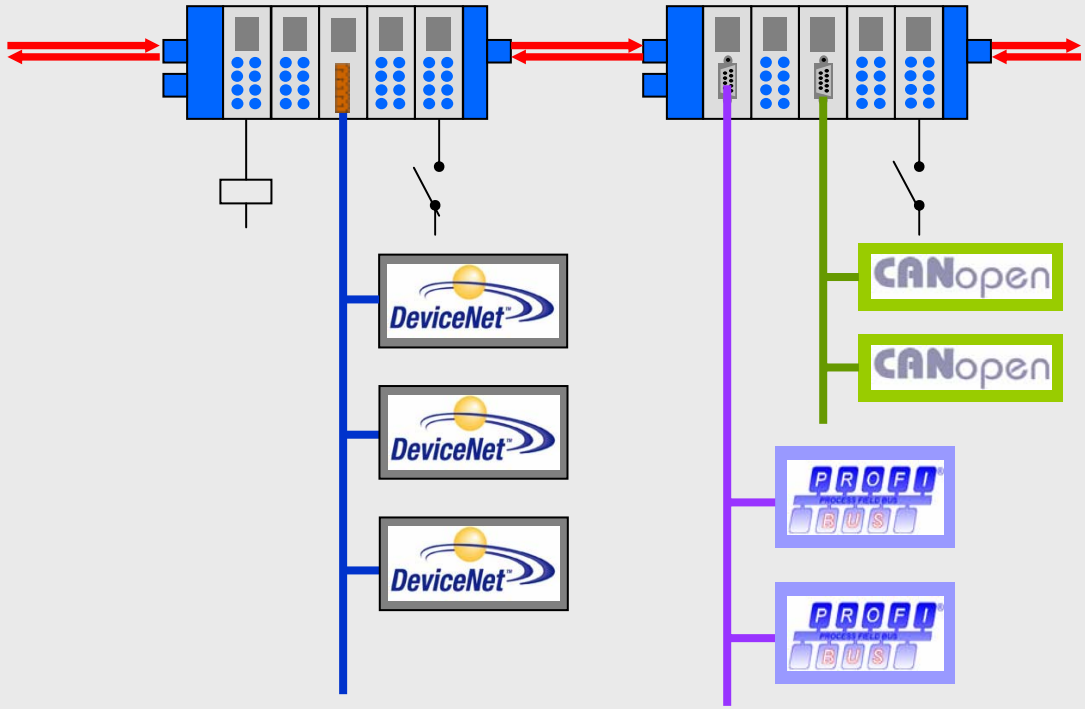
BK1120 + DI 4 + DO 8 + AO 2



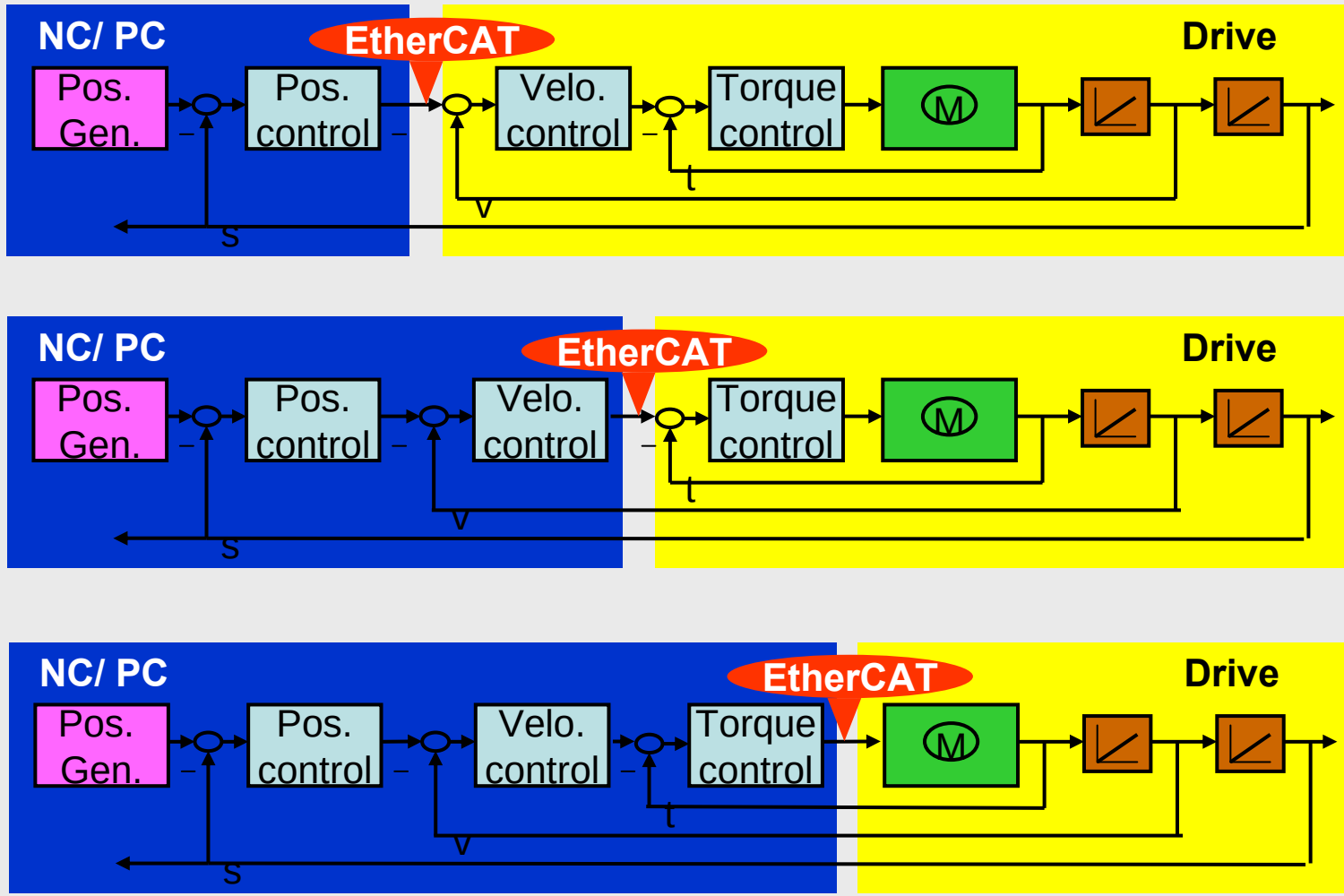
SI…Subindex (SI0 = number of SIs of this object)

Fieldbus Gateway Profile

- Gateways from EtherCAT to legacy Fieldbusses like
  - CANopen
  - Profibus
  - DeviceNet
  - …



Drive Control - Modes of Operation

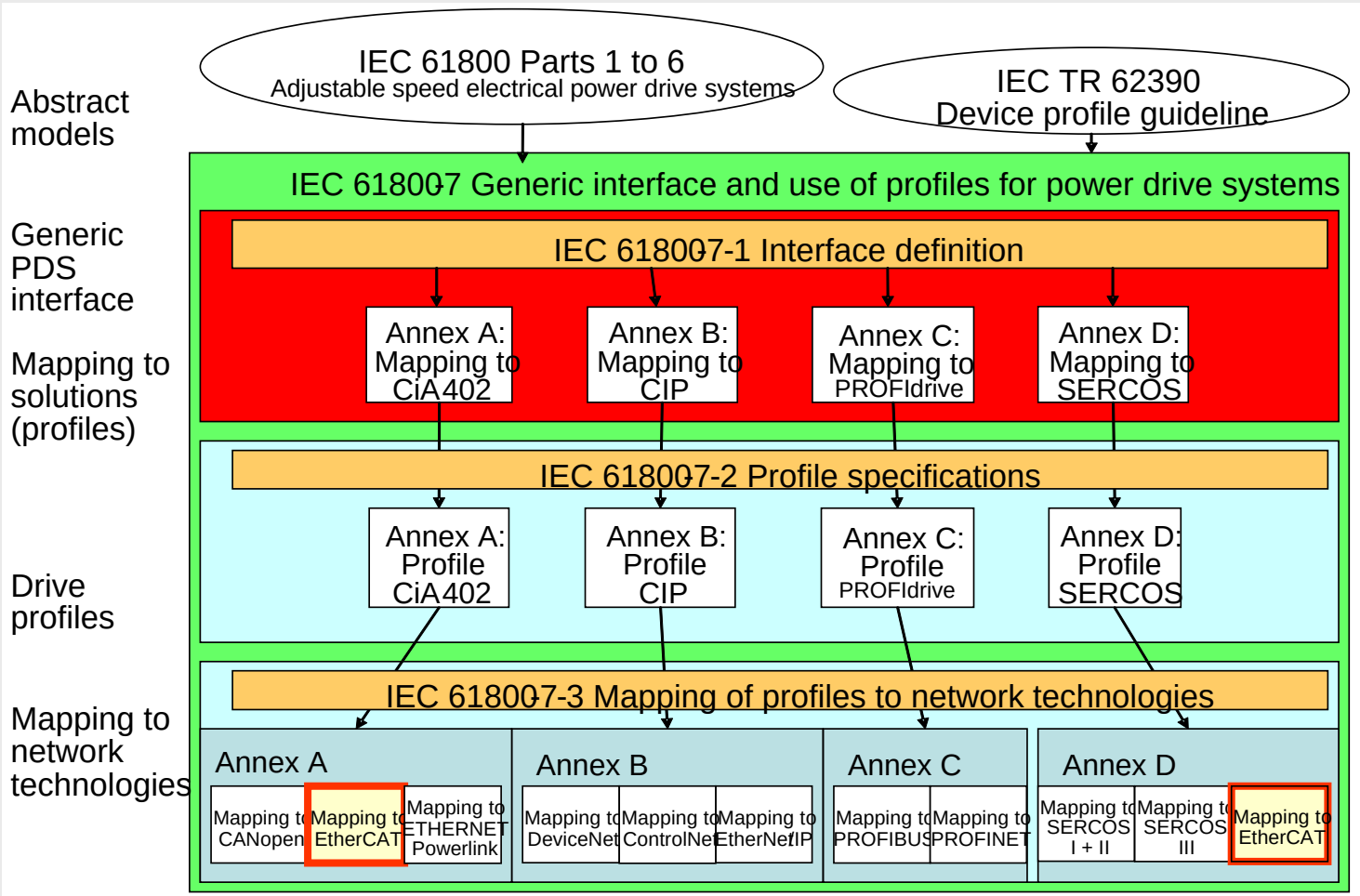


Purpose of Drive Profiles

- Definition of drive specific parameters
- Mapping of the parameters
- State machine of drives
- Synchronization behaviour

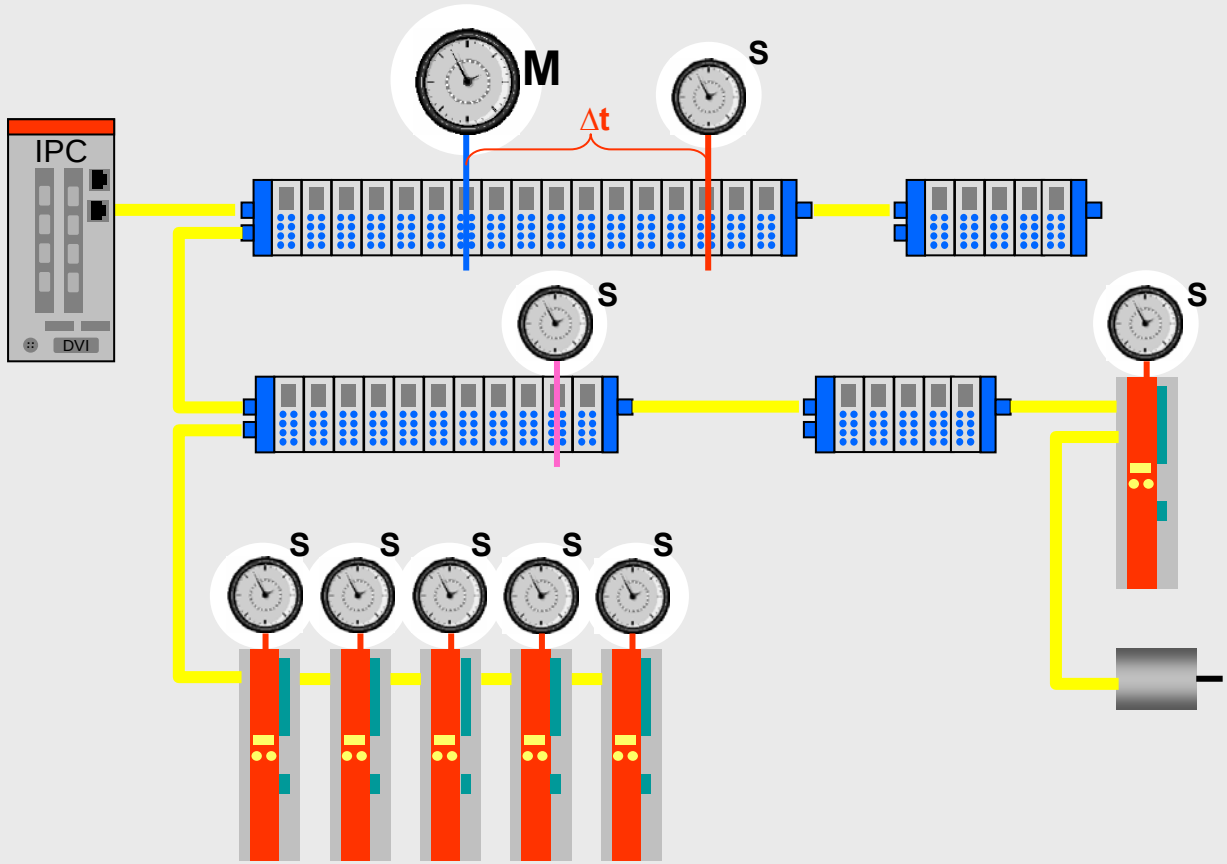
Drive Profile

EtherCAT is Part of the drives standard in Annex A and D



Purpose of Distributed Clocks (DC)

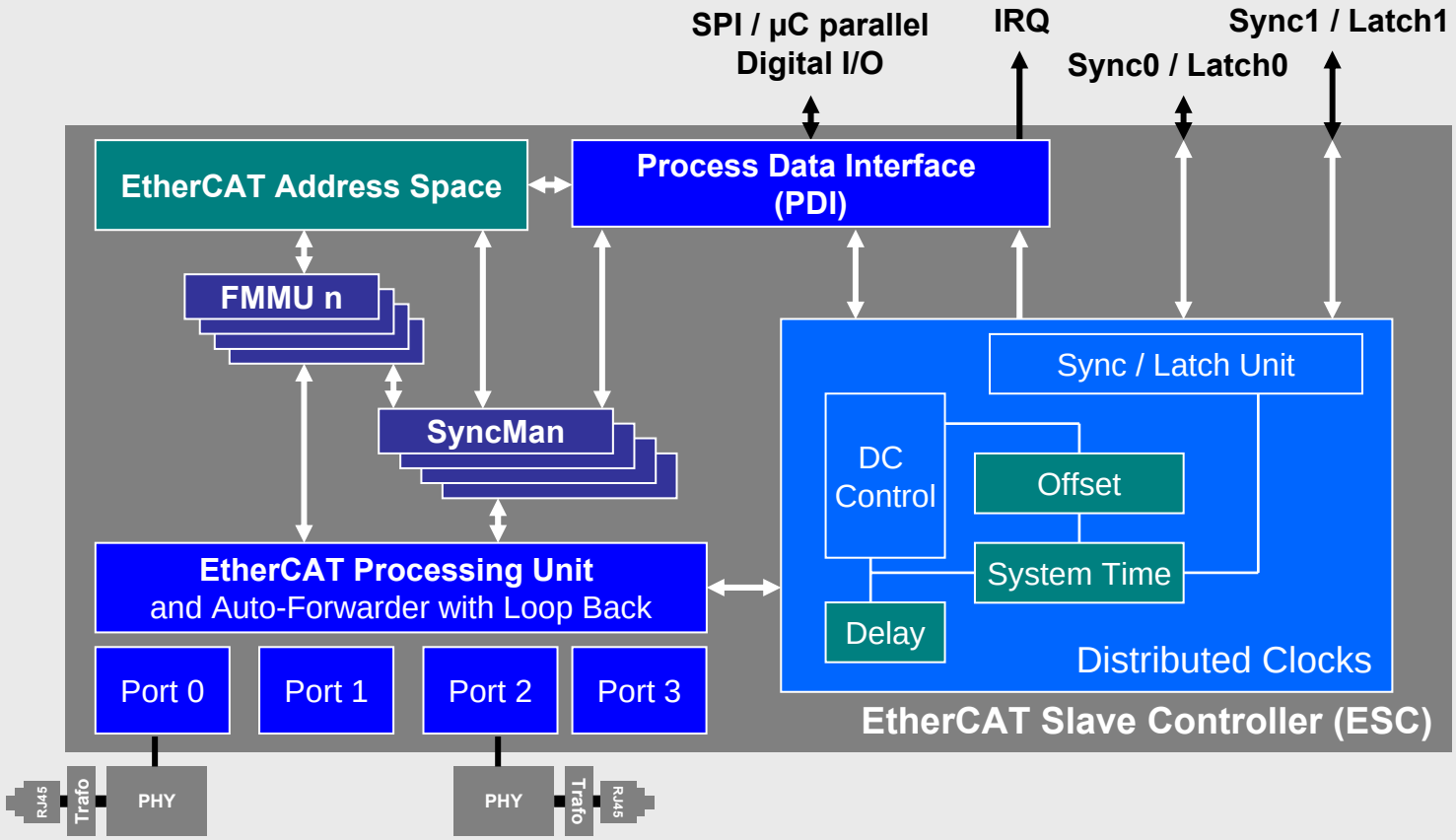
Precise Synchronization (<< 1 µs!) by exact adjustment of distributed clocks



Distributed Clocks – Features

- Synchronization of EtherCAT devices
- Definition of a System Time
  - Beginning on January, 1st 2000 at 0:00h
  - Base unit is 1 ns
  - 64 bit value (enough for more than 500 years)
  - Lower 32 bits spans over 4.2 seconds
    - Normally enough for communication and time stamping
- Definition of a Reference Clock
  - One EtherCAT Slave will be used as a Reference Clock
  - Reference Clock distributes its Clock cyclically
  - Reference Clock adjustable from a “global” Reference Clock – IEEE 1588

Distributed Clocks Unit



Features of DC Unit within ESC (I)

- Provider for local time signals
  - Generation of synchronous local output signals (SYNC0, SYNC1 Signals)
  - Generation of synchronous Interrupts
- Synchronous Digital Output updates and Input sampling
- Precise time stamping of input events (Latch unit)

Features of DC Unit within ESC (II)

- **Propagation delay** measurement support
  - Each EtherCAT slave controller measures the delay between the two directions of a frame
  - Master calculates the propagation delays between all slaves
- **Offset compensation** to Reference Clock
  - Offset between local clock and Reference Clock
  - Same absolute system time in all devices
  - Simultaneousness (clear below 100ns difference) in all devices
- **Drift compensation** to Reference Clock
  - DC Control Unit

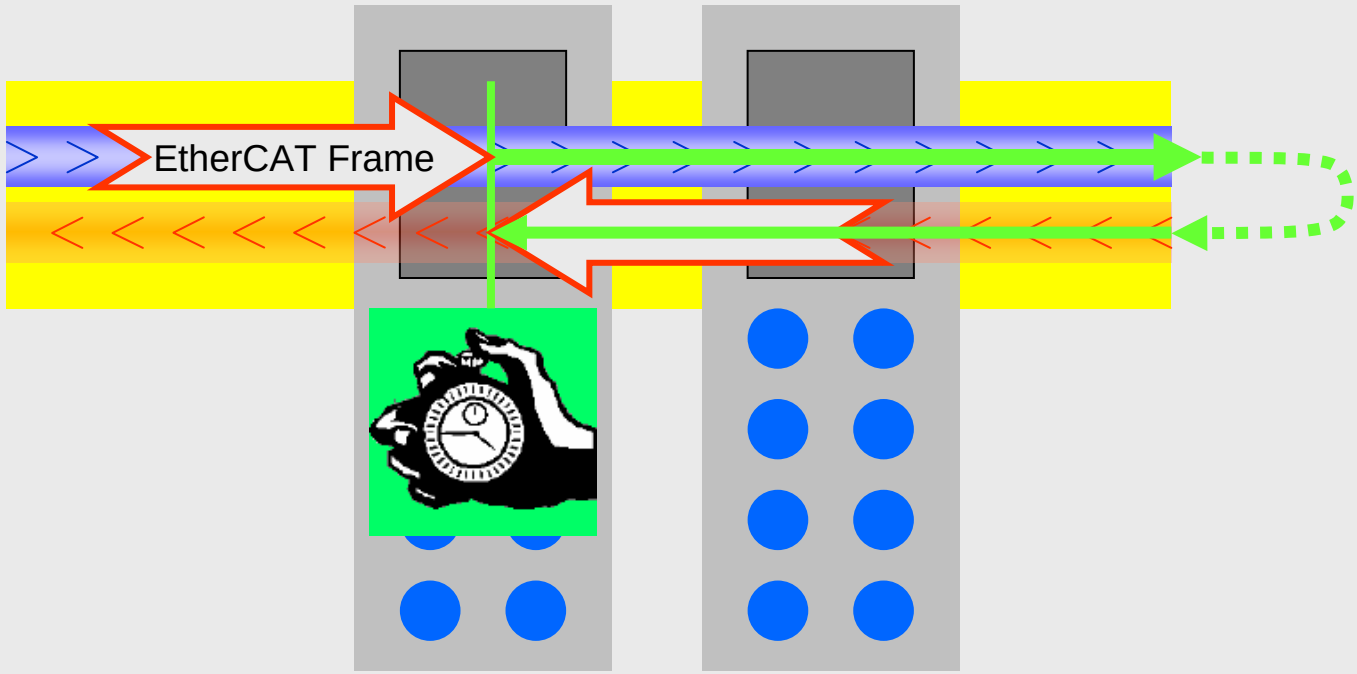
Register System Time

- Registers:
  - *System Time*
    (0x0910:0x0917, small systems 0x0910:0x0913)

- *System Time*
  - Read access from both sides (EtherCAT and µC)
  - Consistent access from µC
    (access to first byte saves an internal copy)
  - Consistent access from EtherCAT
    (within a single frame, internally latched with SOF)
  - Write access from EtherCAT starts the DC control
  - ARMW command (auto increment read – multiple write) allows to read *System Time* of the reverence clock and write it to all slave clocks within a single frame

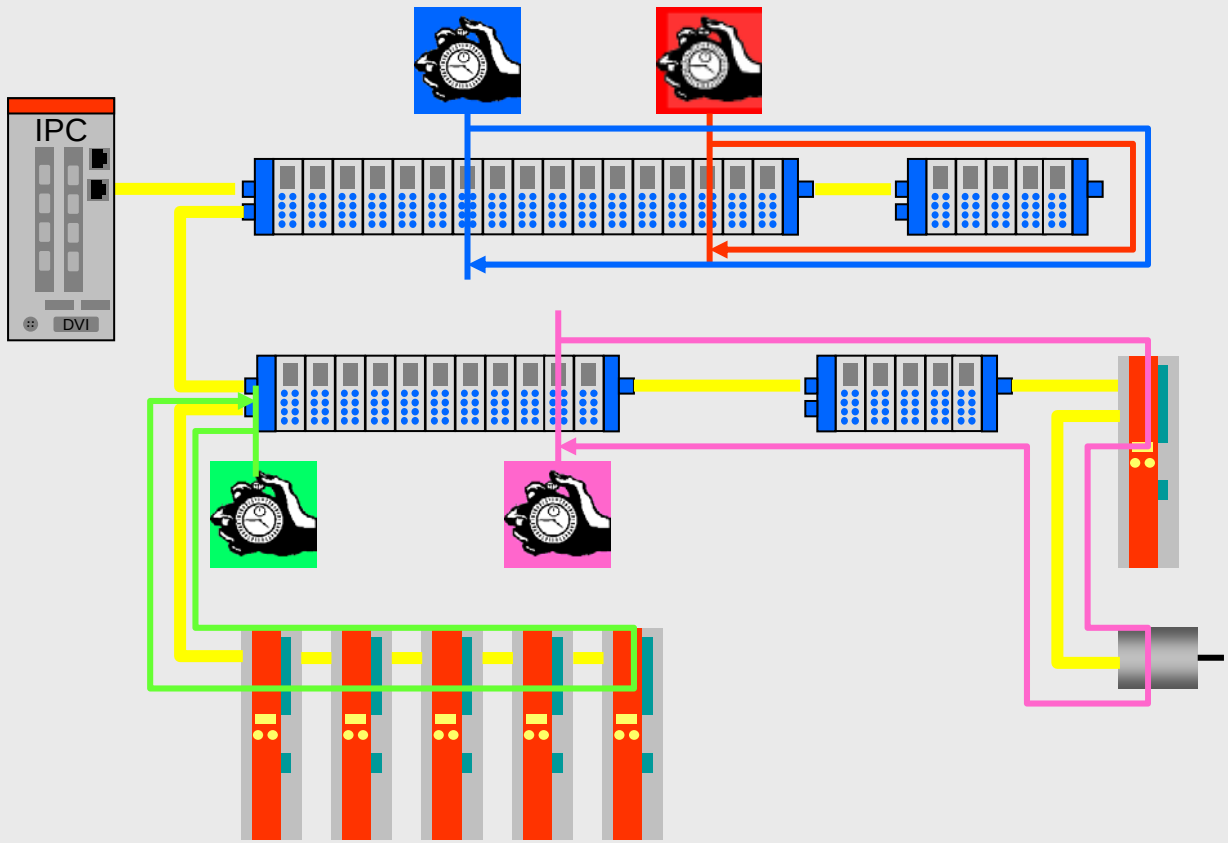
DC – Propagation Delay Measurement (I)

EtherCAT Node measures time difference between leaving and returning frame



DC – Propagation Delay Measurement (II)

Propagation delays between any nodes can be computed



Propagation Delay Measurement

- Registers:
  - *Receive Time Port 0* (0x0900:0x0903)
  - *Receive Time Port 1* (0x0904:0x0907)
  - *Receive Time Port 2* (0x0908:0x090B)
  - *Receive Time Port 3* (0x090C:0x090F)
  - *System Time Delay* (0x0928:0x092B)

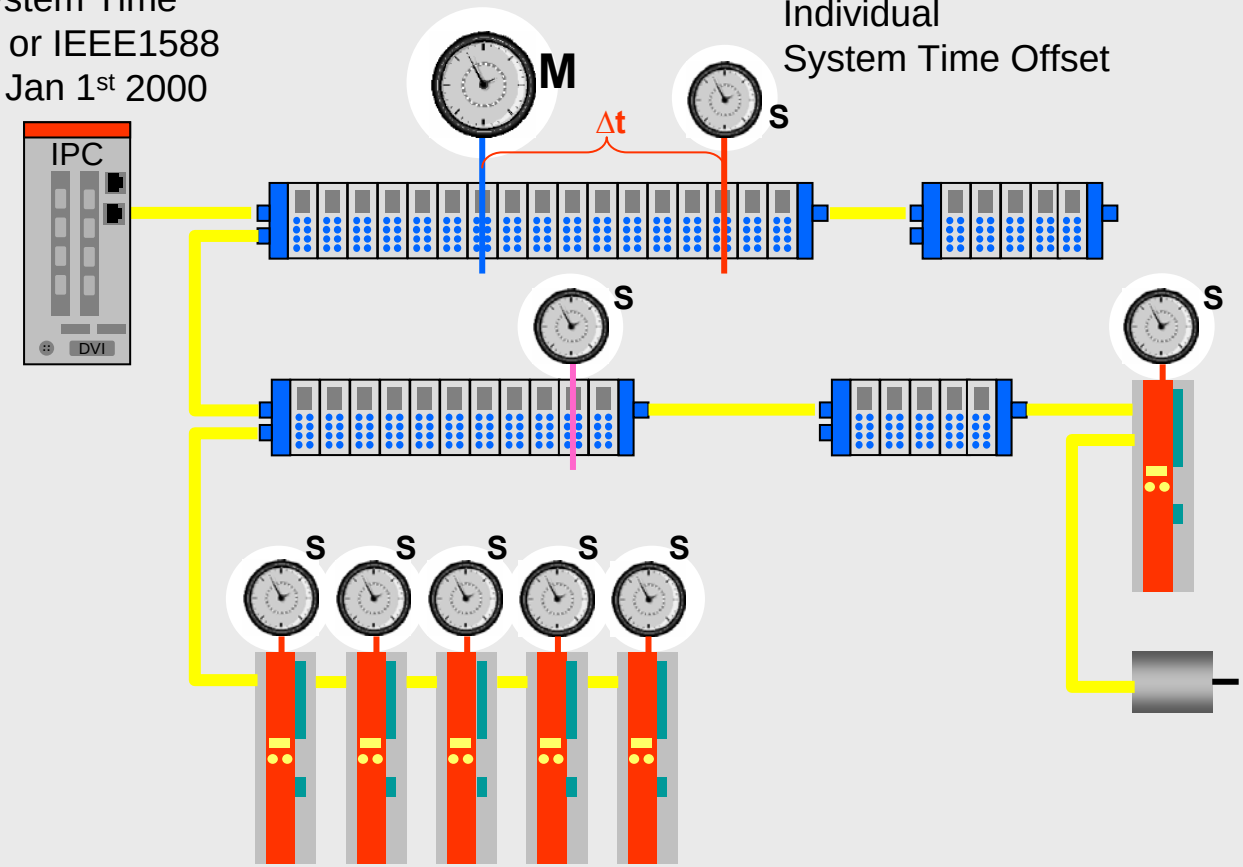
- Write access to *Receive Time Port 0* activates latch
  - Latch local time of SOF (Start of Frame)
  - At EOF (End of Frame) SOF time is copied to *Receive Time Port X*
- *Receive Time Port X* in local clock units (controlled)
- SOF time of all frames are latched on all ports internally
- Master reads all time stamps and calculates the delay times with respect to the topology.
- Individual delay time is written to register *System Time Delay*

Offset and Drift Compensation

Same System Time in all devices

Global System Time
e.g. by RTC or IEEE1588
Reference: Jan 1st 2000

Individual
System Time Offset



Offset Compensation

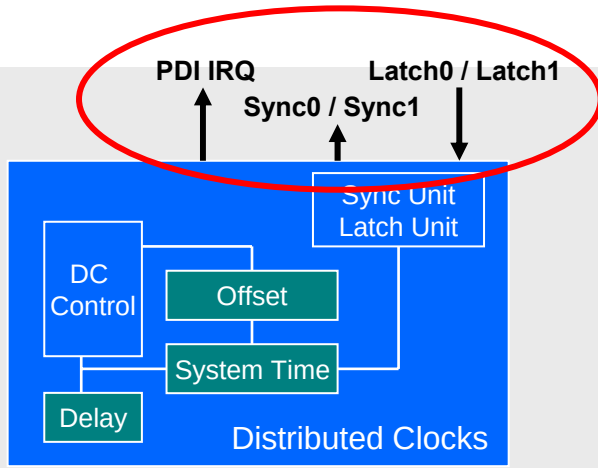
- Registers:
  - *System Time Offset*
    (0x0920:0x927, small systems 0x0920:0x0923)

- Difference between the Reference Clock and every slave device's clock is calculated by the master.
- The offset time is written to register *System Time Offset*
- Each slave calculates its local copy of the System time using its local time and the local offset value:
  - $t_{\text{Local copy of System Time}} = t_{\text{Local time}} + t_{\text{Offset}}$

Drift Compensation – DC Control

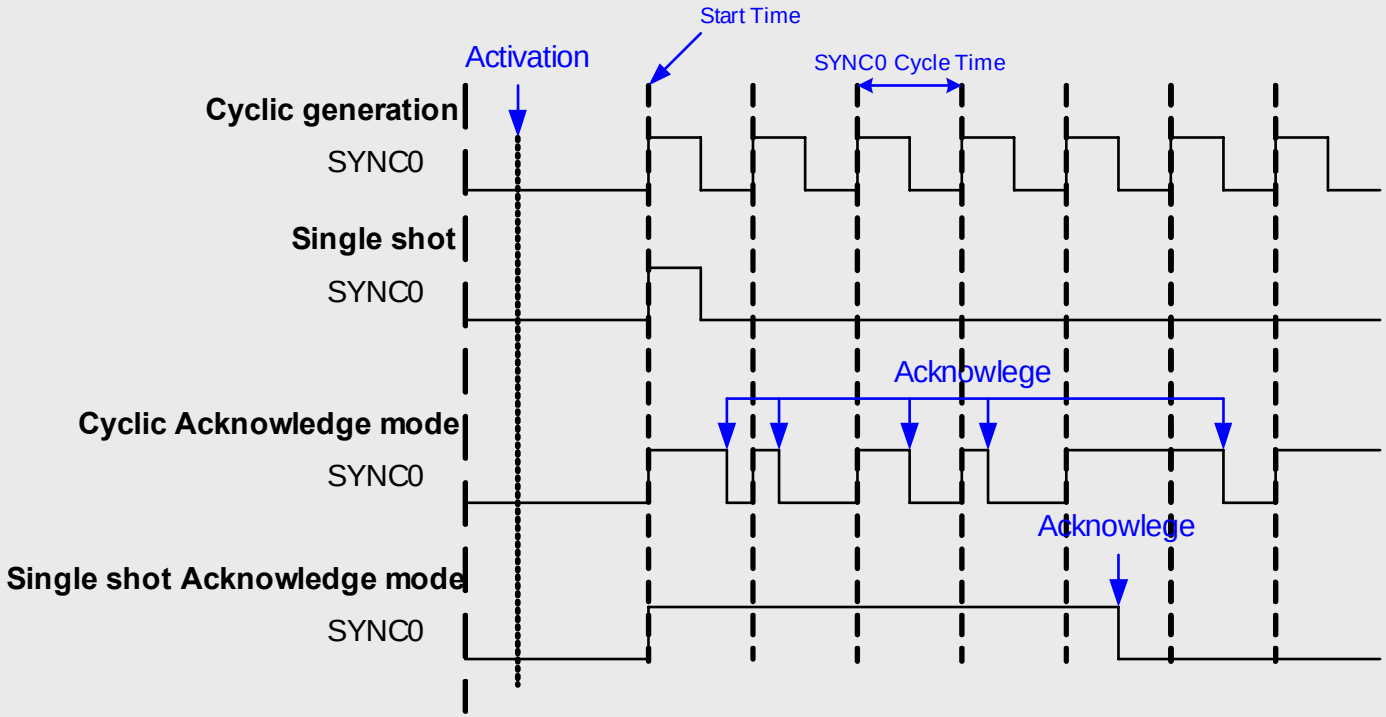
- Registers:
  - System Time (0x0910:0x0917, small systems 0x0910:0x0913)
  - System Time Offset (0x0920:0x927, small systems 0x0920:0x0923)
- ARMW command (auto increment read – multiple write) allows to read *System Time* of the reverence clock and write it to all slave clocks within a single frame
- DC Control
  - Write access to *System Time* compares received Time with local time
    - $\Delta t = (t_{Local\ time} + t_{Offset} + t_{PropagationDelay}) - t_{Received\ Time}$
  - If ($\Delta t > 0$) then decelerate local clock
    else if ($\Delta t < 0$) accelerate local clock

SyncSignal Generation



- Output of the Sync unit can be used for
  - Interrupt generation
  - PDI Digital Output Update events
- Can be mapped to
  - *AL Event Request* Register for PDI IRQ
  - SYNC0 and SYNC1
- SyncSignals can be generated at a specific System Time
- Four Operation modes are supported:
  - Cyclic generation
  - Single shot
  - Cyclic acknowledge
  - Single shot acknowledge
- The second SyncSignal (SYNC1) depends on SYNC0, it can be generated with a predefined delay after SYNC0 pulses
- Initiated alternatively from the EtherCAT master or slave application side

SYNC0 Signal Generation modes



Pulse Length of SYNC Signals (0x0982:0x0983)	SYNC0 Cycle Time (0x09A0:0x09A3)	
	> 0	= 0
> 0	Cyclic generation	Single Shot
= 0	Cyclic Acknowledge*	Single Shot Acknowledge*

* Acknowledge by reading SYNC status register (0x098E, 0x098F)

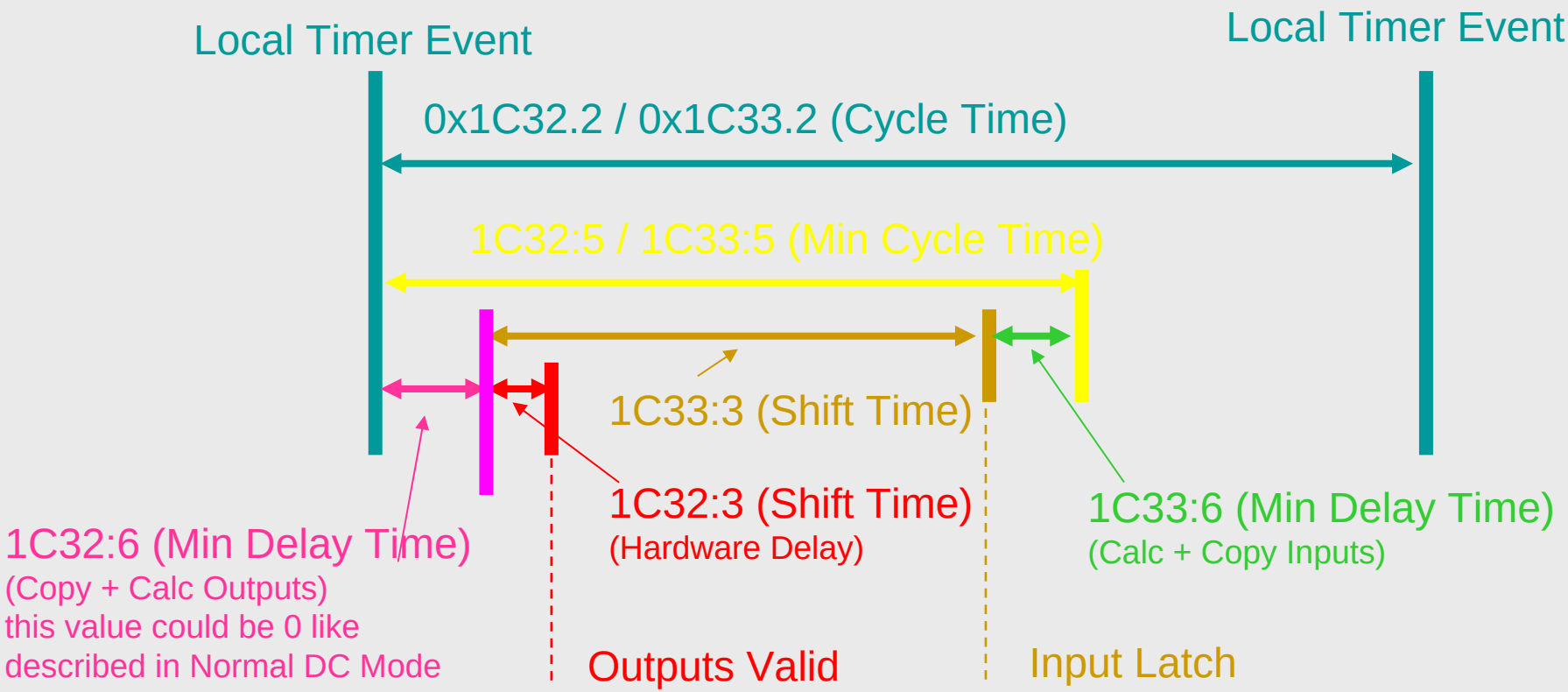
Latch Functionality

- The Latch Event unit supports time stamping of the system time with two independent input signals
  - LATCH0 and LATCH1 input signals are used
    (can be the same pins as for SYNC0 and SYNC1, ESC dependent)
  - Time Stamping of SyncManager events is possible
- Latch on positive and/or negative edge
- Single or continuous latch configurable
- The *Latch Time* register (0x09B0:0x09CF) contain the time stamps
  - Acknowledge by reading the *Latch Time* register.

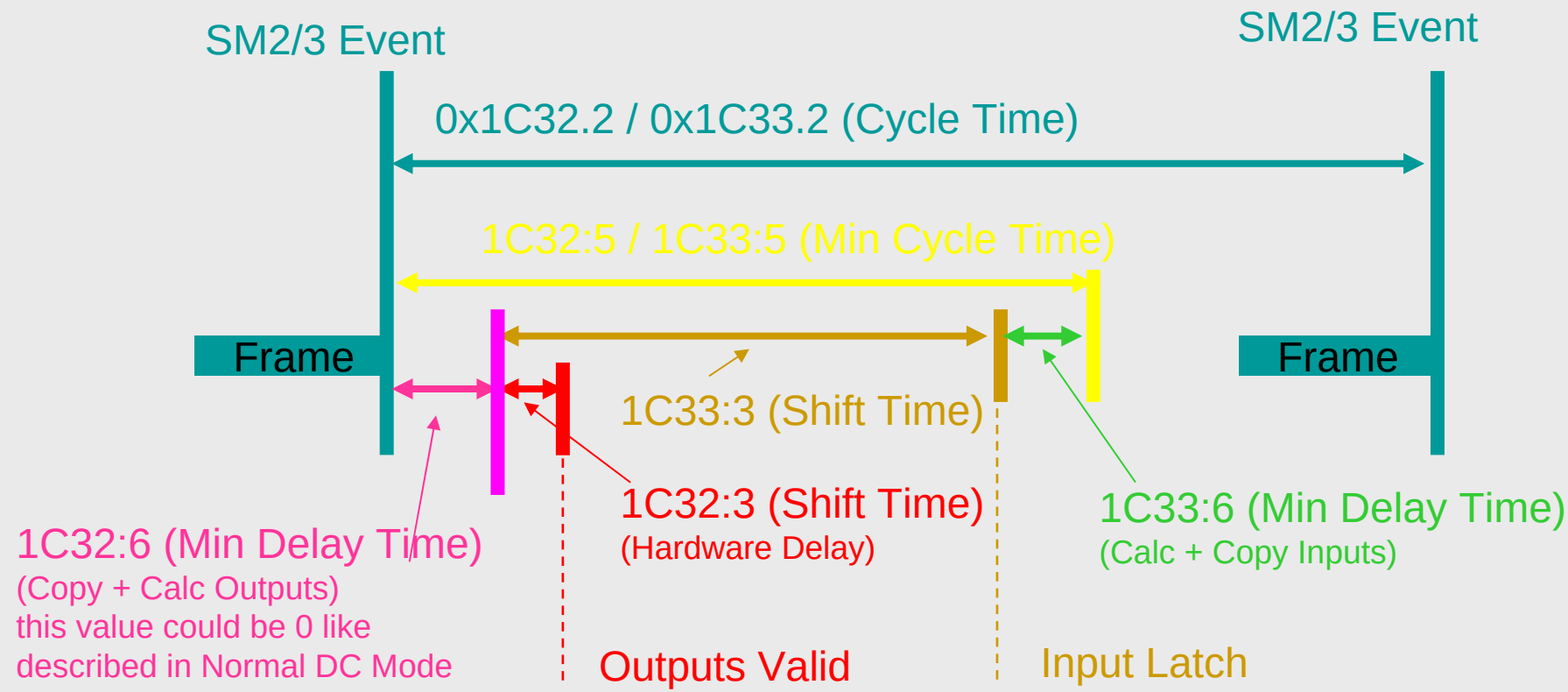
Local Synchronization Modes in the Slaves

- Free Run
  - Slave's application is not synchronized to EtherCAT
- Synchronous with SM Event
  - Slave's application is synchronized to the SM2 Event (if cyclic outputs are transmitted) or the SM3 Event (if only cyclic inputs are transmitted)
- Synchronous with SYNC Event
  - Slave's application is synchronized to the SYNC0 or SYNC1 Event
    - Normal Mode
    - Optimized Mode

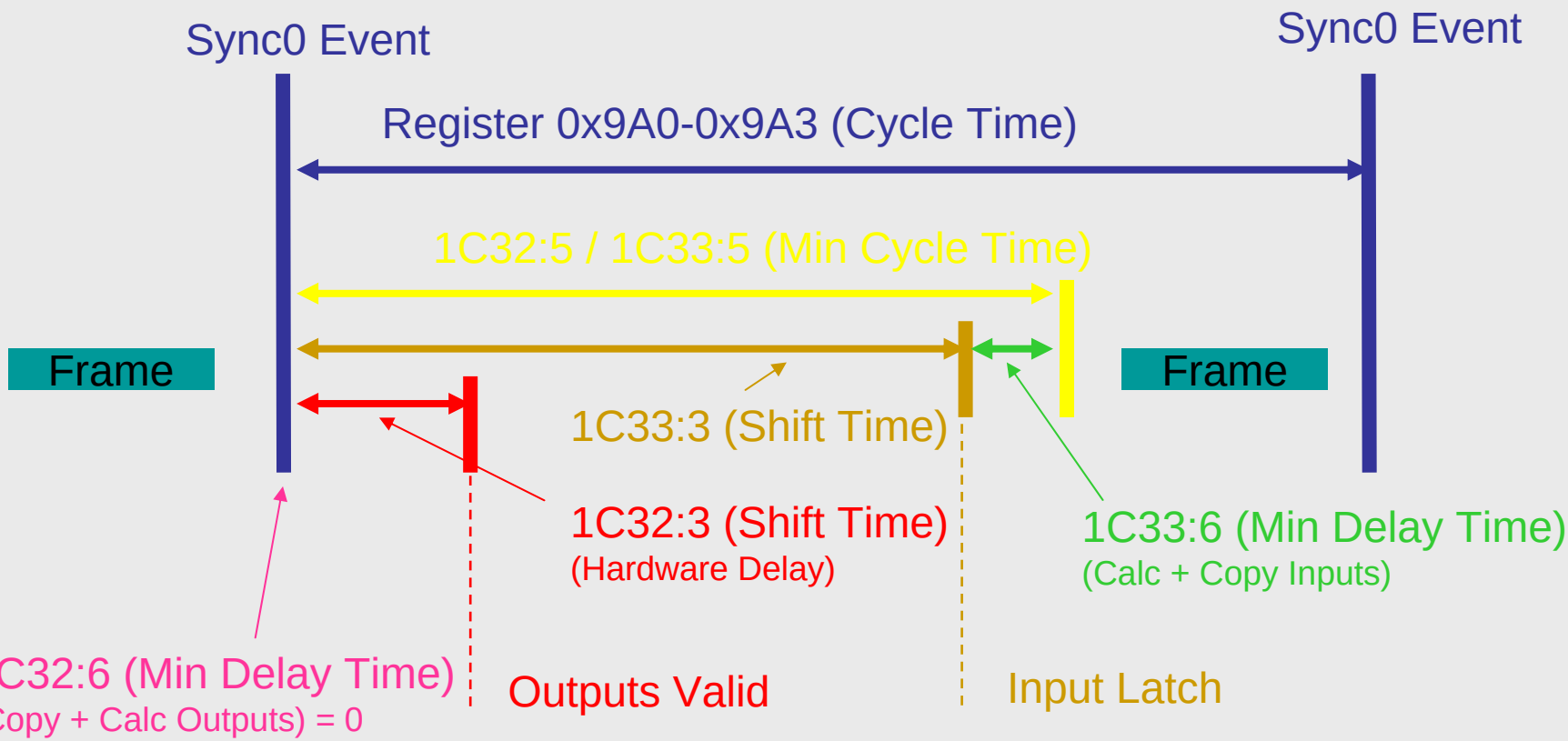
Free Run


Local Timer Event
Local Timer Event
0x1C32.2 / 0x1C33.2 (Cycle Time)
1C32:5 / 1C33:5 (Min Cycle Time)
1C33:3 (Shift Time)
1C32:3 (Shift Time)
(Hardware Delay)
1C33:6 (Min Delay Time)
(Calc + Copy Inputs)
1C32:6 (Min Delay Time)
(Copy + Calc Outputs)
this value could be 0 like
described in Normal DC Mode
Outputs Valid
Input Latch

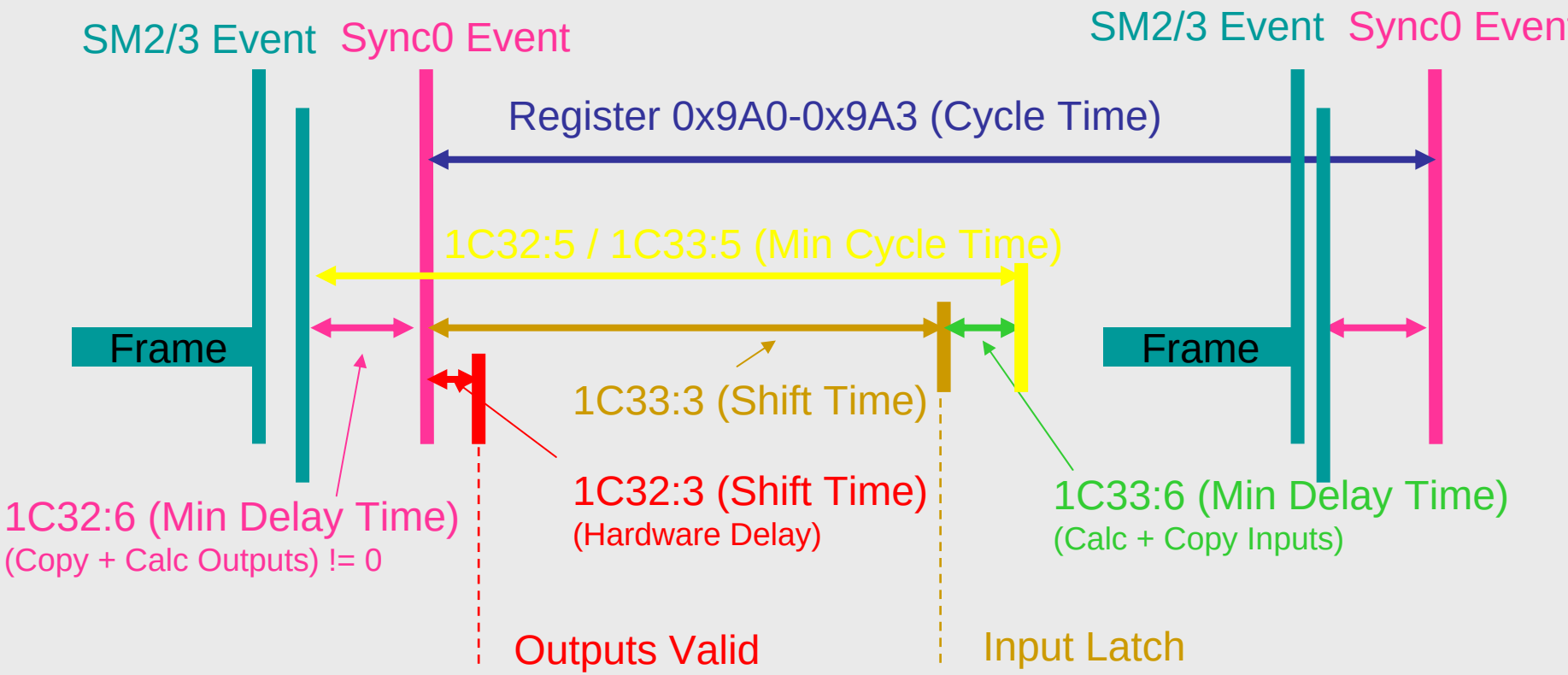
Synchronous to SM2/3 event



Normal DC mode



Optimized DC mode



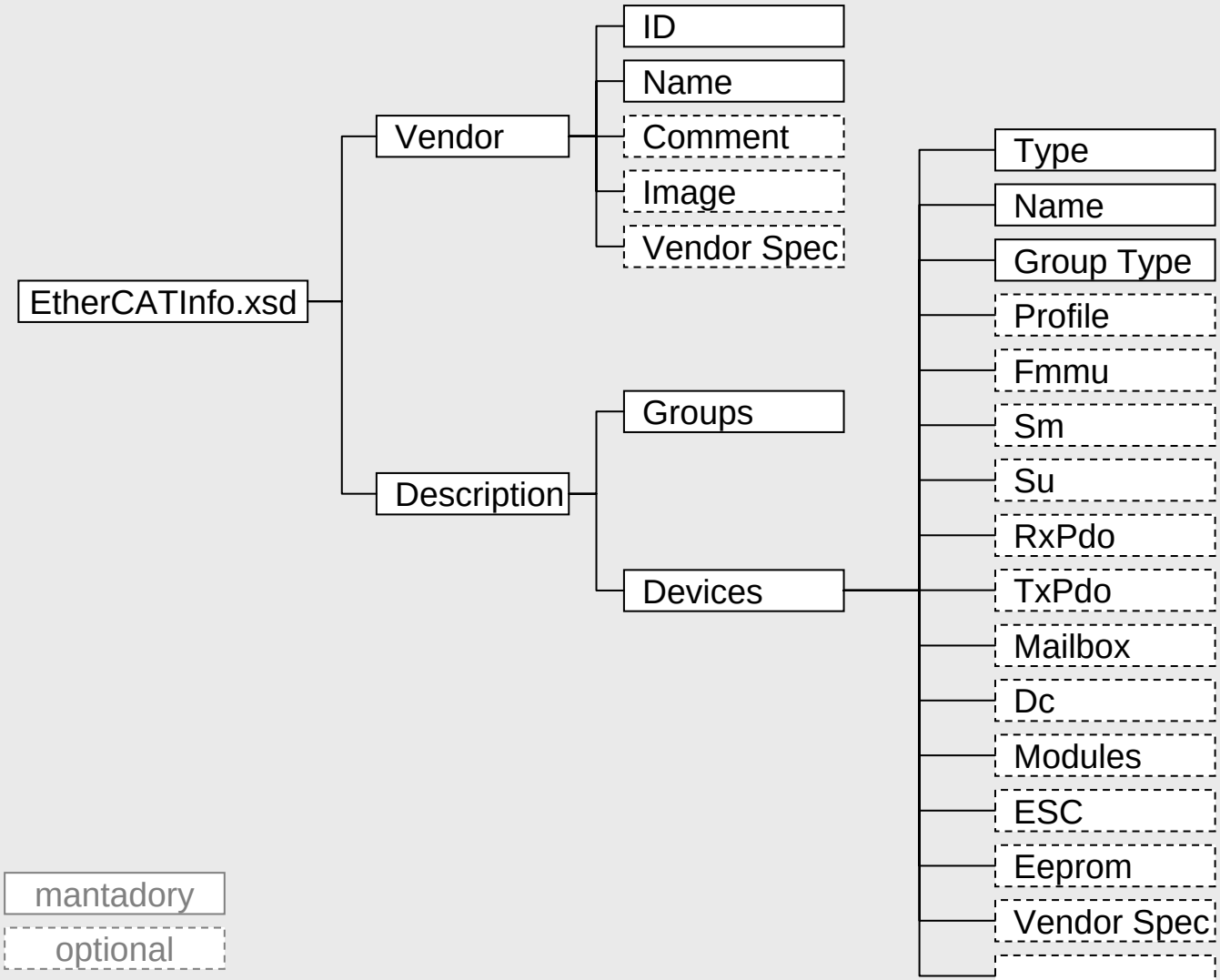
Device Description Overview

- Device Description File in XML format
- One file suitable for a set of devices (from one Vendor)

- File contains information about:
  - Vendor
    - Vendor ID, Name, Logo, …
  - Device groups
    - Organization units to help configuration tools
  - Device
    - Device Identity, Name, PDI type
  - PDO Mapping
  - FMMU / SyncManager
    - number and usage

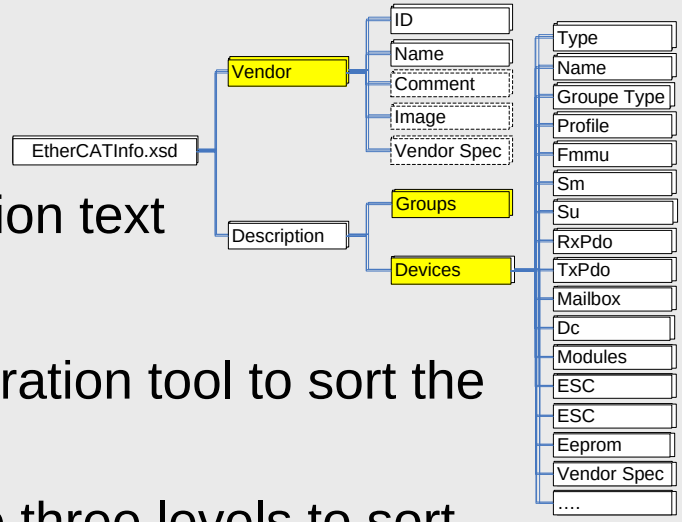
- Schema is defined in “EtherCATInfo.xsd”
- Can be viewed with Browser, Text Editor or XML Editors

General Structure

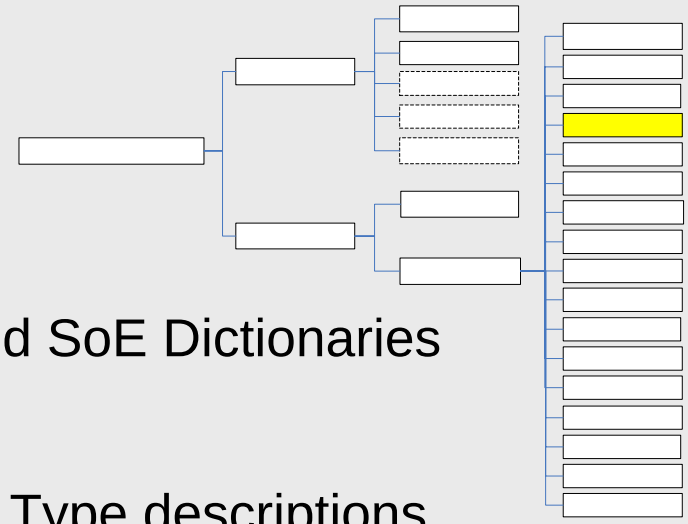


Attributes *Vendor, Groupes, Devices*

- Vendor
  - vendor ID, vendor name
  - Bitmap image and information text
- Groups
  - Device groups help configuration tool to sort the devices.
  - Configuration tool may use three levels to sort devices: vendor, group and device.
  - At least one device group must be provided, each device is assigned to one group
- Devices
  - Device description shall contain Type, Name and Group Type
  - FMMU, SM, PDO, Mailbox and Eeprom are recommended



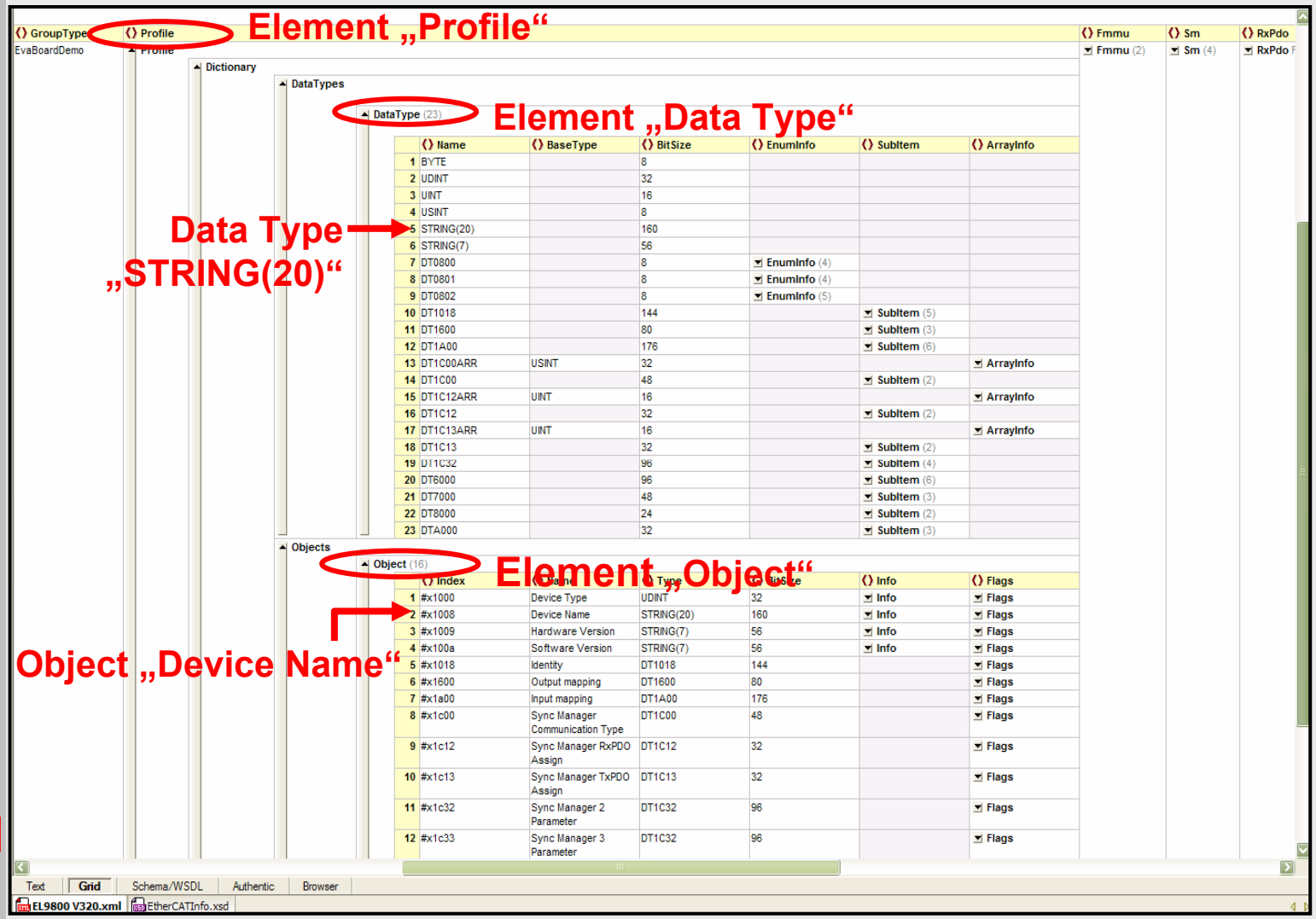
Attributes *Profile*



- Profile
  - Profile Information
  - Object Dictionary
  - Same Structure for CoE and SoE Dictionaries

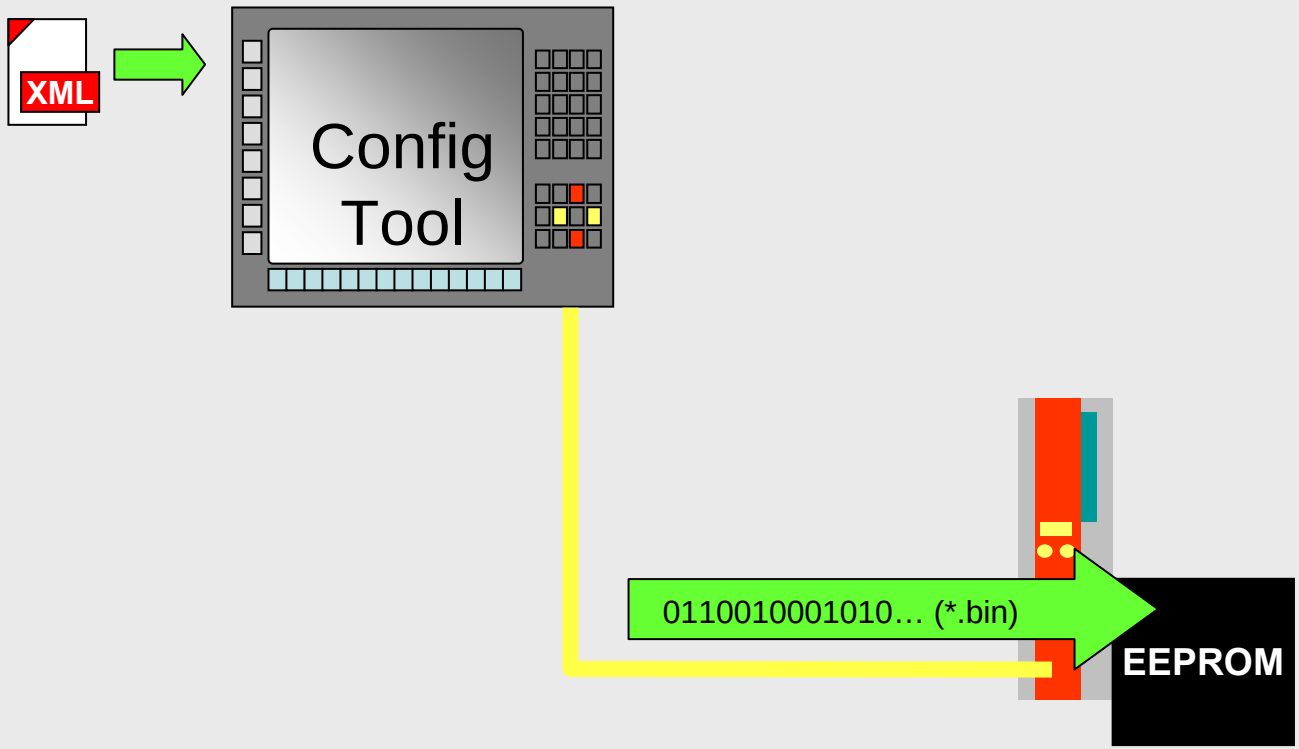
- Separation of Object and Data Type descriptions
  - Flat list of Objects
  - All Objects derived from a Data Type
  - ARRAY, RECORD (CoE) and Variable Data (SoE) defined as Data Types
  - Data Type describes the complete data of an object exactly as for a “Complete Access” download or upload

Data Types and Objects



XML and EEPROM

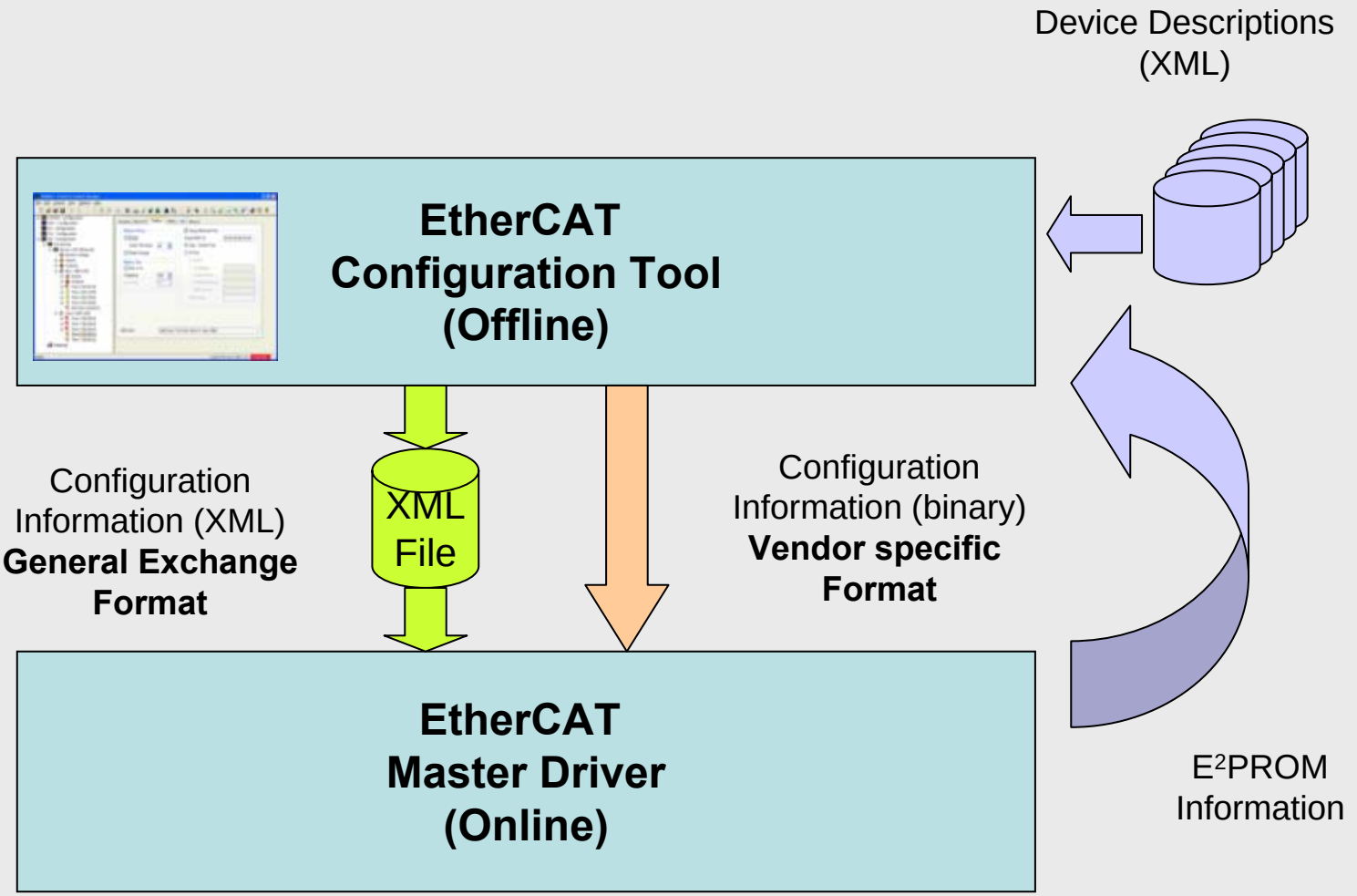
Configuration Tool generates binary file from device description to update EEPROM on slave



EtherCAT Configuration Tool

- Configure EtherCAT Slave devices
  - Evaluate XML device description
  - Evaluate EEPROM information – if online
- Generate network initialization commands
  - Information for the EtherCAT driver
  - Initialization commands correspond to State Machine transitions
- Generates cyclic commands
  - Information for the EtherCAT driver

Network Configuration



EtherCAT Configuration Exchange Format

- Vendor and Driver independent format
- Master Vendor must not imperatively develop an own Configuration Tool
- Contains
  - initialization commands – per slave device
  - cyclic process data commands
  - information about the mailboxes

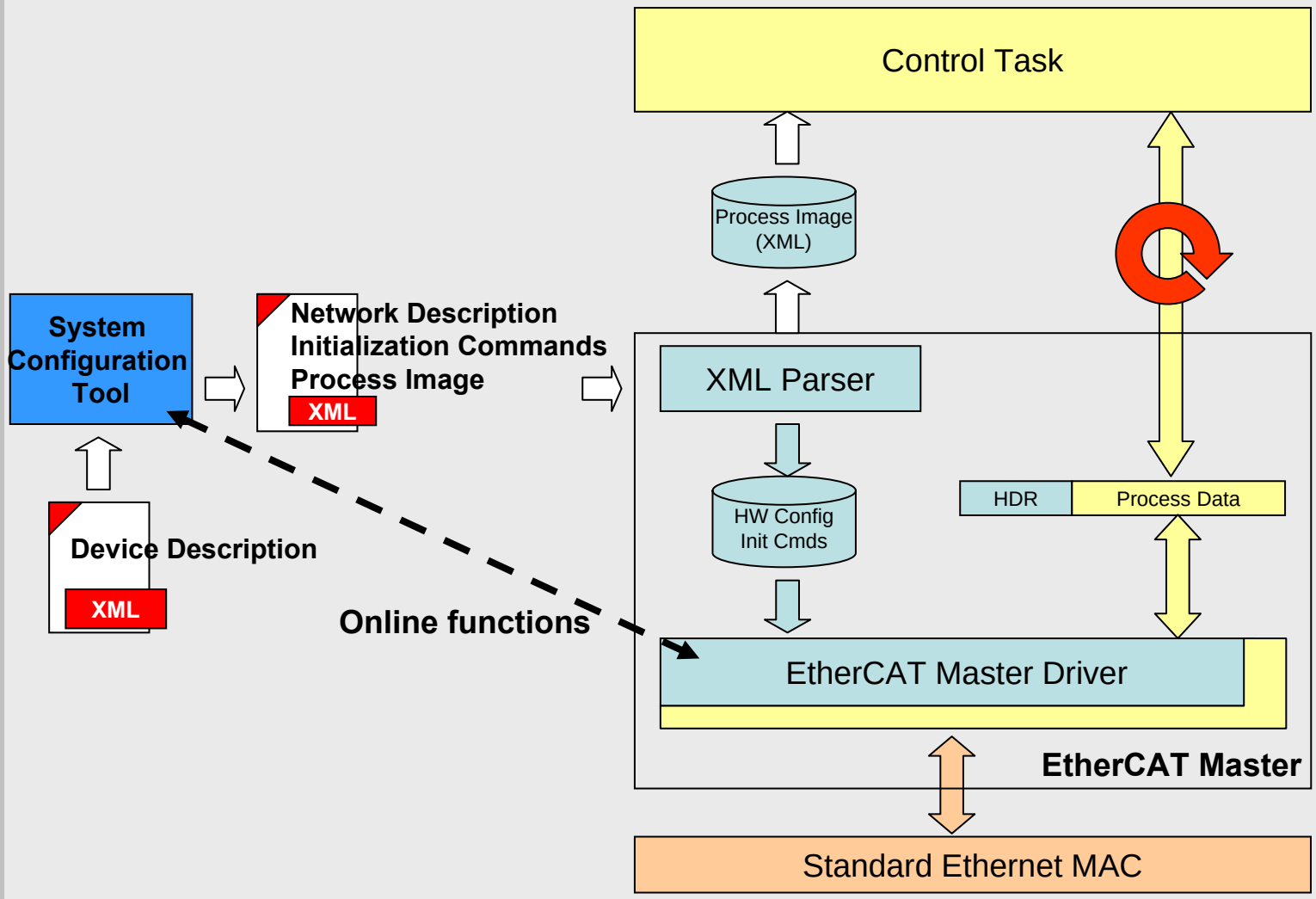
EtherCAT Master

- Configuration with help of an EtherCAT configuration XML file
- Send and receive raw Ethernet frames from a network adapter
- Management of the EtherCAT slaves
  - Sending init commands defined in the XML file
- Mailbox Communication
  - CANopen over EtherCAT protocol (CoE)
  - Servo-Profile over EtherCAT protocol (SoE)
  - Ethernet over EtherCAT protocol (EoE)
  - Filetransfer over EtherCAT protocol (FoE)
- Software-integrated switch functionality
- Cyclic process data communication

What does an EtherCAT Master do?

- Parse XML Hardware configuration file (initialization, state machine, and process data mapping)
- Initialization of Fieldbus
- Runs State Machine
- Interface to application
- Interface to network driver
- Sends cyclic process data commands
- Sends mailbox commands
- Handles various protocols

EtherCAT Master Block Diagram



Prerequisites for EtherCAT Master (Real Time)

- **Hardware**
- Standard network controller using DMA
  - **NO** special plug in card needed
  - Speed and Quality important
- No switches or hubs required
- Cache design, CPU
- Low jitter, x86 Dual Xeon < 2 µs

- **Software**
- Realtime kernel
- Configuration tool

EtherCAT Master Development (Real Time)

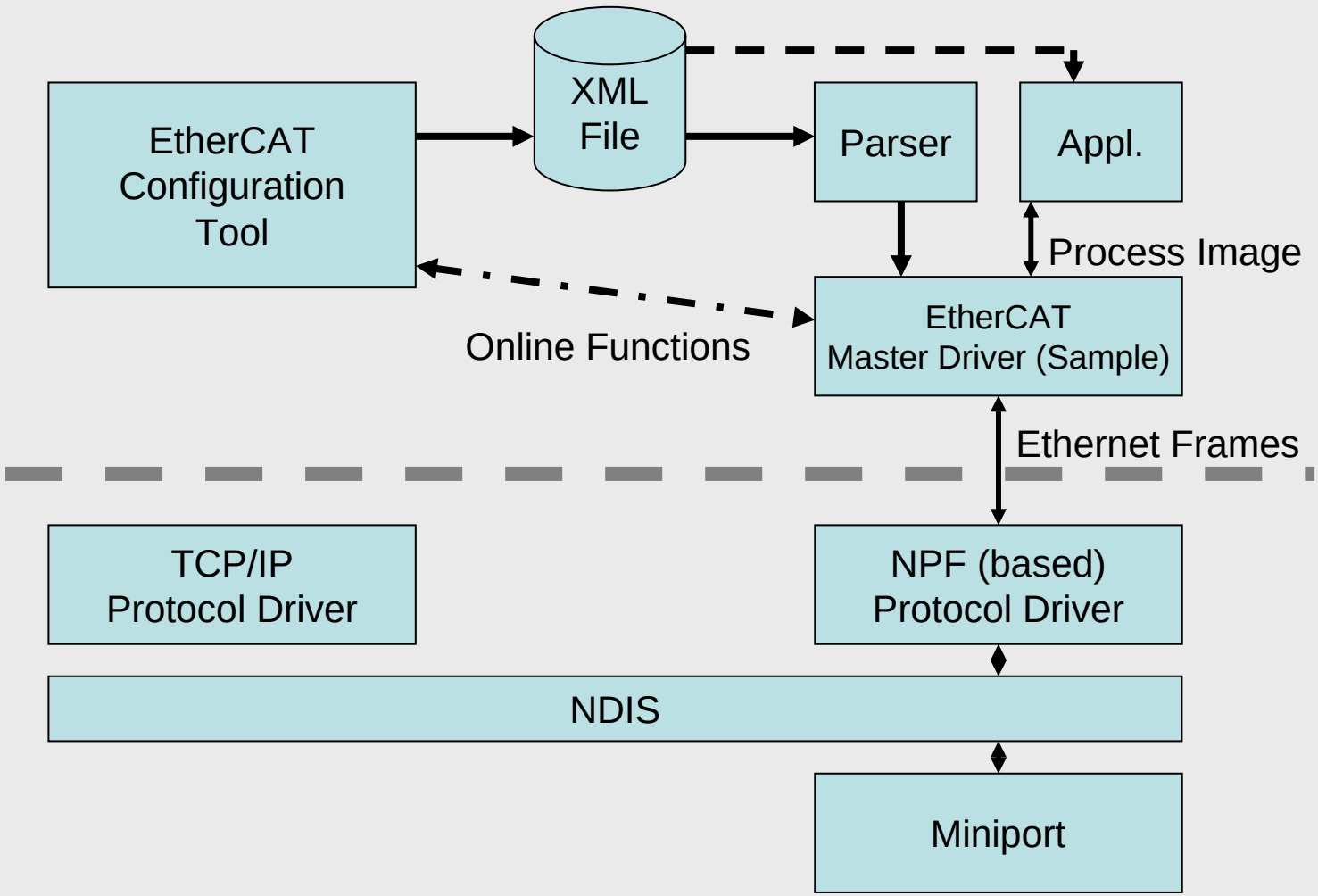
1. XML Parser functionality
2. EtherCAT Master driver
   - Interface to configuration tool
   - State machine
   - Interface to application
   - Interface to network card
   - NIC Timing Interface
3. Real Time Kernel

necessary

4. Hardware Configuration Tool
   - 3.rd party configuration tools can be used as the configuration is provided to EtherCAT master in a common format (XML)

optional

Master Sample Code Structure



Master Sample Code Overview

- Sample EtherCAT Master Communication Software (including Source Code)
  - Non Real Time
  - Realized as Windows Application Program (MS Windows XP/2000)
  - Source Code MS C++

Standards and References

- Specification of EtherCAT has been done in the EtherCAT Technology Group (ETG)
- Specifications available at www.EtherCAT.org
  - XML File Style sheet
  - Datasheets of ESC, ...
  - Modular Device Profile
  - Reports of ETG TC meetings
- International standardization efforts

Standards and References

- EtherCAT is part of different international standardization efforts

Standard	Title	Status	Remarks
IEC/PAS 62407	Real Time Ethernet control automation technology (ETHERCAT)	PAS	
IEC 61158	Digital data communication for measurement and control – Fieldbus for use in industrial control systems		Type 12: EtherCAT Specification
	Part 1: Overview and guidance	CDV	
	Part 2: Physical Layer service definition and protocol specification	CDV	
	Part 3: Data Link Layer service definition	CDV	
	Part 4: Data Link Layer protocol specification	CDV	
	Part 5: Application Layer service definition	CDV	
	Part 6: Application Layer protocol specification	CDV	

PAS : Public available standard
CDV : Committee draft for vote

Standards and References

Standard	Title	Status	Remarks
IEC 61784	Digital data communication for measurement and control		
	Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems	CDV Ed2.0	
	Part 2: Additional profiles for ISO/IEC 8802-3 based communication networks in real-time applications	CDV Ed1.0	CPF12: EtherCAT
IEC 61800	Adjustable speed electrical power drive systems		
	Part 7-1: Generic interface and use of profiles for power drive systems – Interface definition	CDV	Part 7: Drive Profiles
	Part 7-2: Generic interface and use of profiles for power drive systems – Profile specifications	CDV	
	Part 7-3: Generic interface and use of profiles for power drive systems – Mapping of profiles to network technologies	CDV	Mapping EtherCAT to CANopen DS402 and SERCOS

CPF : Communication profile

Standards and References

Standard	Title	Status	Remarks
ISO 15745	Industrial automation systems and integration -- Open systems application integration framework	Ed 1	CANopen
	Part 4 Amd 2: Profiles for Modbus TCP, EtherCAT and ETHERNET Powerlink	PRF Amd	Mapping EtherCAT to CANopen DS301

PRF Amd: : Proof of a new International Standard, Amendment

IEC 61158

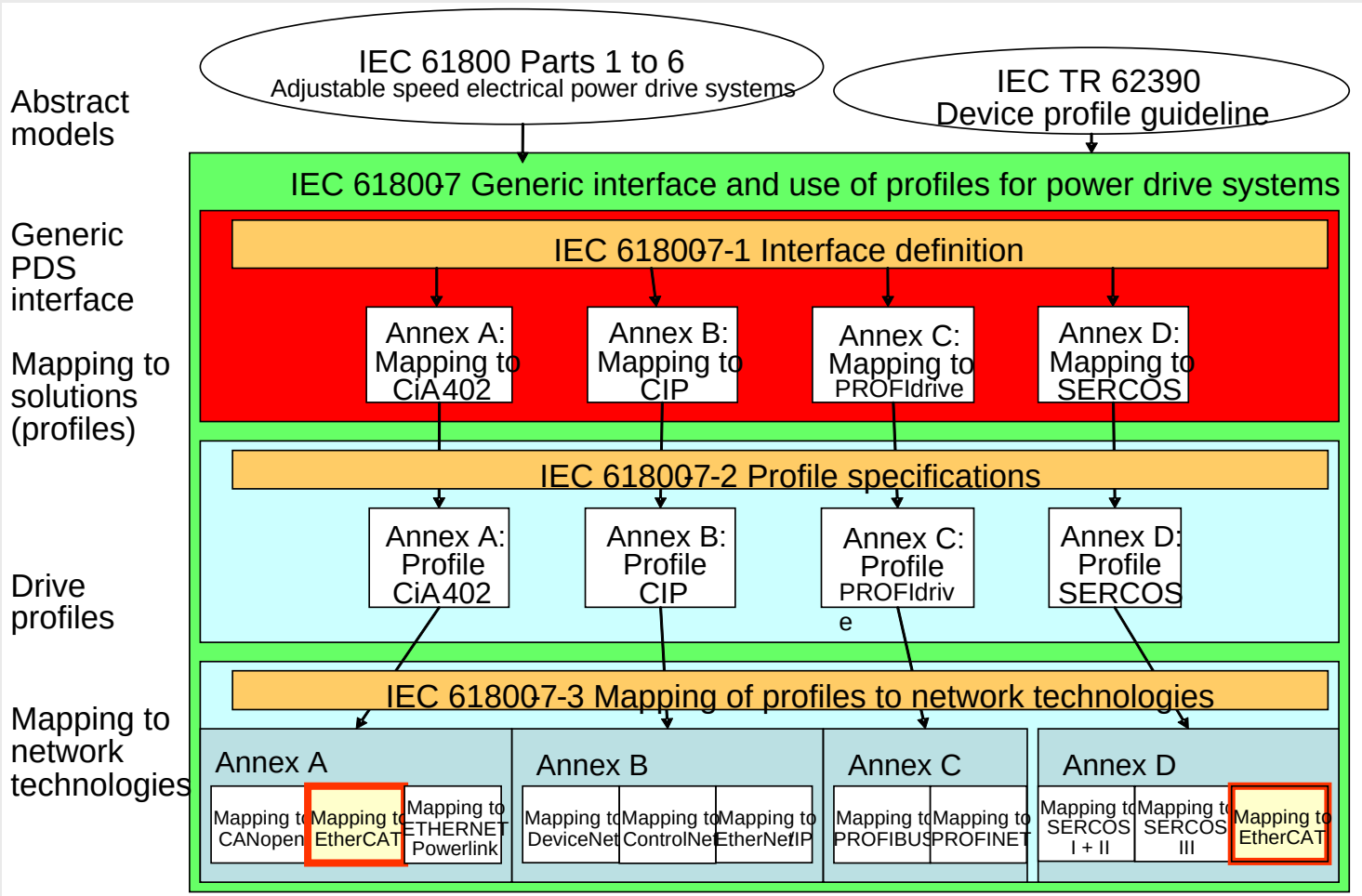
- Digital data communication for measurement and control – Fieldbus for use in industrial control systems
- *The* communication standard
- EtherCAT is named Type 12 in IEC 61158 (no brand names allowed)
- Transformation of the communication protocol to a common model

IEC 61158 – DL/AL services and protocols

Users	Implementers
AL Services Part 5 in IEC 61158 • Model and Concepts • Data type definitions • Application Objects • Service description • Communication Management	AL Protocol Part 6 in IEC 61158 • Syntax definition and Coding • Application Relationship Procedures • State Machines
DL Services Part 3 in IEC 61158 • Model and Concepts • Service description • Register Description (DL objects)	DL Protocol Part 4 in IEC 61158 • Coding • Medium Access • State Machines

IEC 61800-7

EtherCAT is Part of the drives standard in Annex A and D



ESC Overview

- Available ESCs:
  - FPGA
  - IP Core
  - ASIC
- Criteria:
  - Features:
    - Number/Type of Ports (MII, EBUS)
    - PDI Type (functionality, speed)
    - RAM
    - DCs, Number of SyncManager, FMMUs
    - …
- Needed Quantity, Flexibility in relation to price

Step by Step Implementation

- **Step 1:**
- Slave Evaluation Kit Workshop

Step 2:

Setup of a small EtherCAT System with Master, I/O and Evaluation Board.

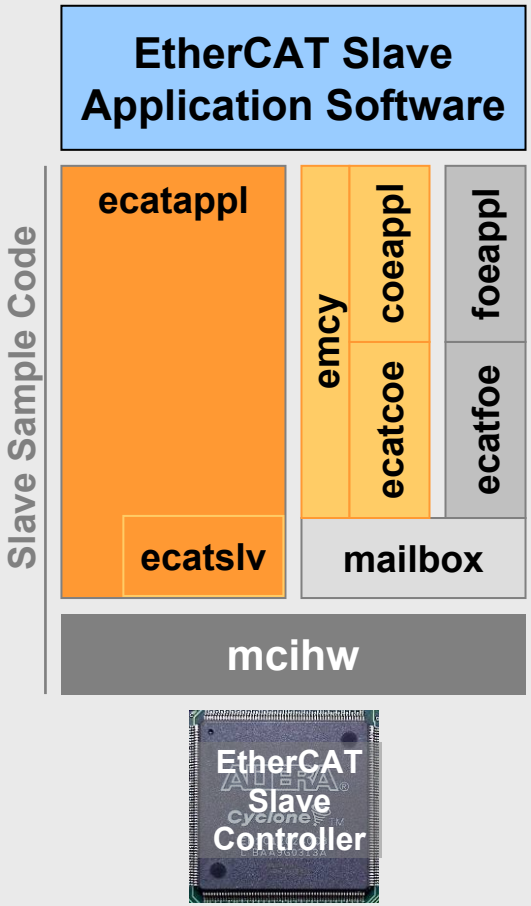
Step 3:

Build Lab wiring between Evaluation Kit and Standard-Hardware (e.g.) with 16Bit Microcontroller

Step 4:

Start SW-Project with Slave Sample Code

Step by Step Implementation



- **Step 5:**
- Download Slave Sample Code in Target Hardware and set System into Operation

Step 6:

Test implementation for PDO- und Mailbox-Transfer

EtherCAT Hardware-Design

Step 7:

Software-Design with EtherCAT Slave Sample Code for existing Fieldbus SW-Architecture