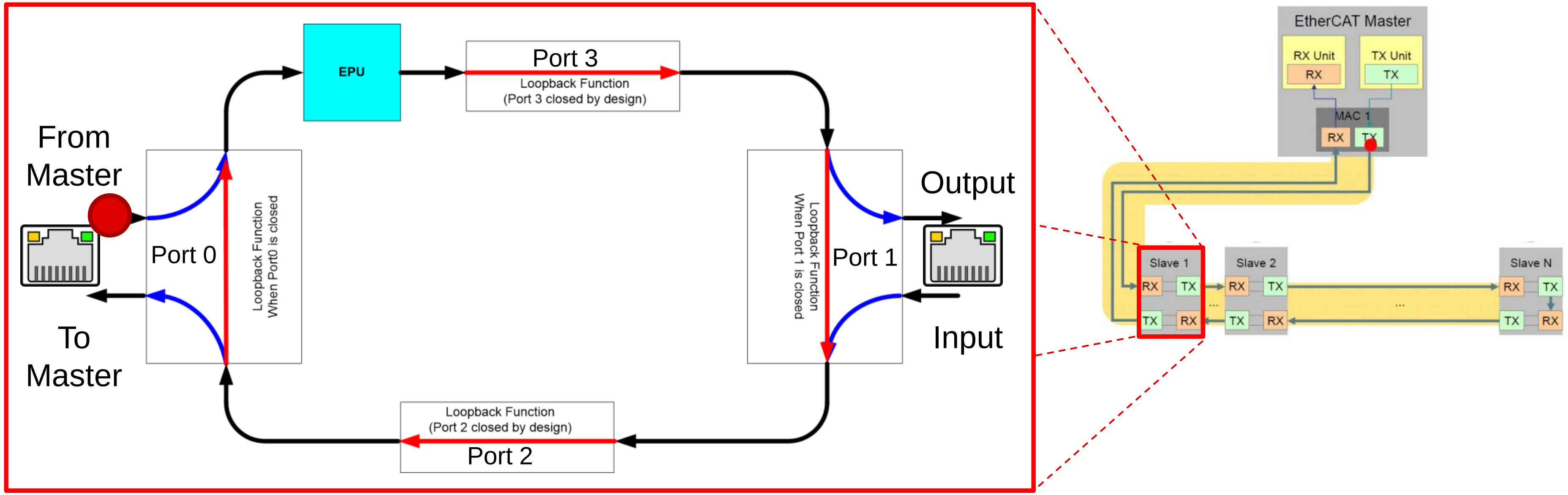
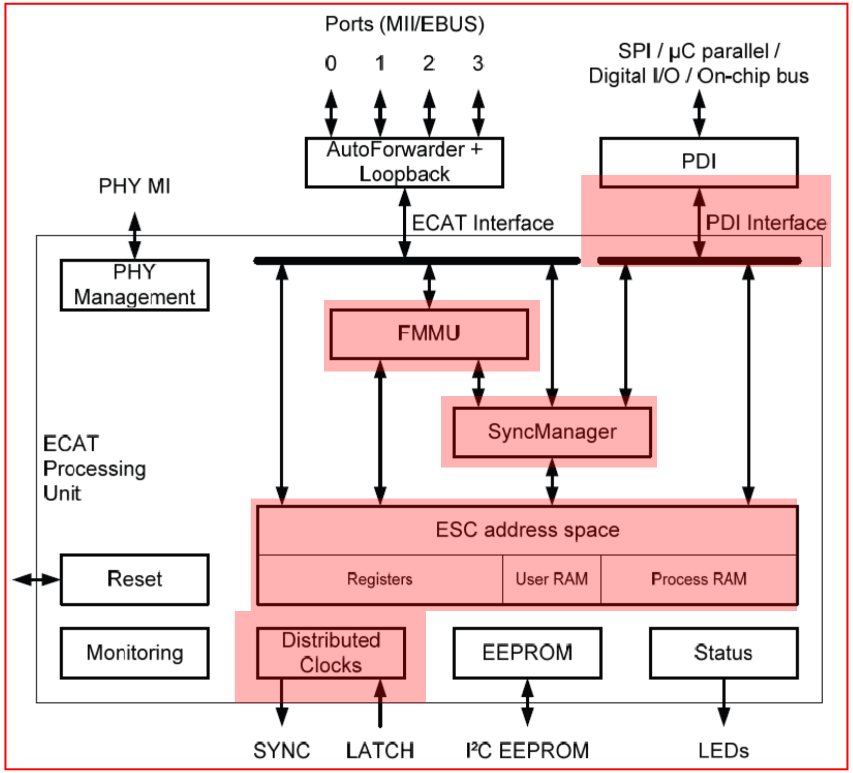
# EtherCAT Slave Overview

# EtherCAT Slave Node


From Master
To Master
Port 0
Loopback Function When Port0 is closed
EPU
Port 3
Loopback Function (Port 3 closed by design)
Loopback Function When Port 1 is closed
Port 1
Output
Input
Loopback Function (Port 2 closed by design)
Port 2
EtherCAT Master
RX Unit
RX
TX Unit
TX
MAC 1
RX
TX
Slave 1
Slave 2
Slave N
RX
TX
...

- For each slave node, data always comes in from the master (Port 0)
- The EtherCAT Processing Unit (EPU) is the logical core of an EtherCAT slave controller. It contains registers, memories and data processing elements. A frame always comes from port A before passing through the EtherCAT Processing Unit. It receives, analyzes and processes the EtherCAT data stream.
- The other ports (1, 2, 3) connect to downstream nodes
- When there is no connection to a port, the port's internal switch closes

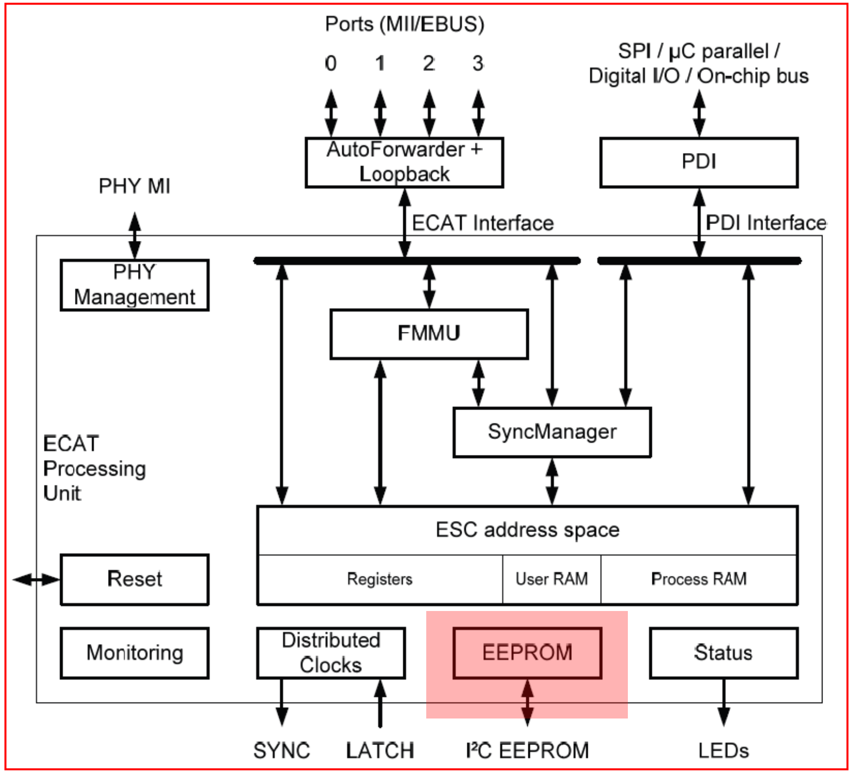
# EtherCAT Slave – EPU Overview (1 of 2)



**Highlights**

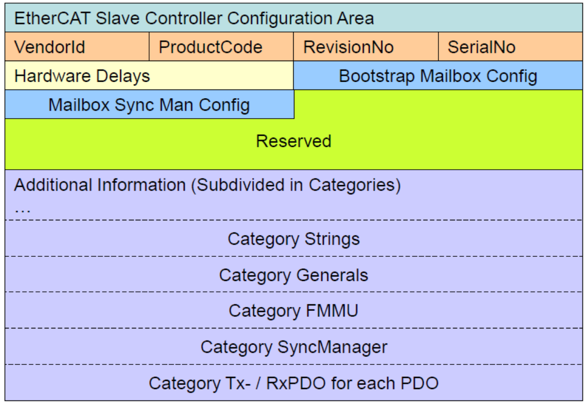
- DPRAM – Each slave contains a Dual-Port RAM that's accessible by the master and slave.
  - Master access is always available.
  - Slave access depends on the state machine state
- FMMU – Memory Management Unit
  - Maps (bitwise!) mapping of logical to physical addresses in the ESC
- SyncManager
  - Manages consistent exchange of data via mailboxes between master and slaves.
- PDI (process data interface)
  - Interface to the device running the protocol stack
  - Example: Via SPI, on-chip bus, EMIF, etc
- Distributed Clock
  - Synchronizes Local clock to Master Reference
  - Provides time/syncronized Input/Output (Sync/Latch)

## EtherCAT Slave – EPU Overview (2 of 2)



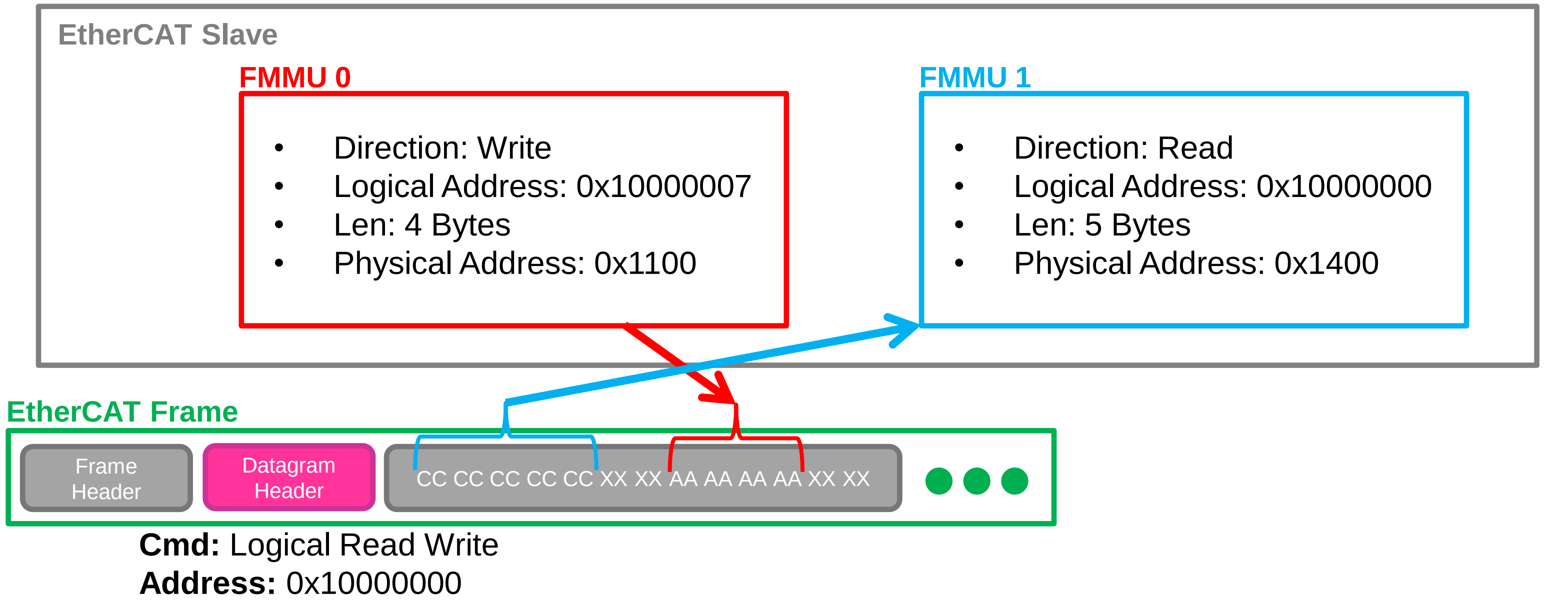
**Highlights**

- EEPROM Interface to standard I2C memories
  - Loaded automatically after ESC Reset
  - Contains Configuration Information including:
    - Slave Node Vendor, Product, Rev/serial #s
    - Communication defaults
    - FMMU & SyncManager data
  - Minimum EEPROM size is 2kbit, 32kb or larger is supported for complex devices.



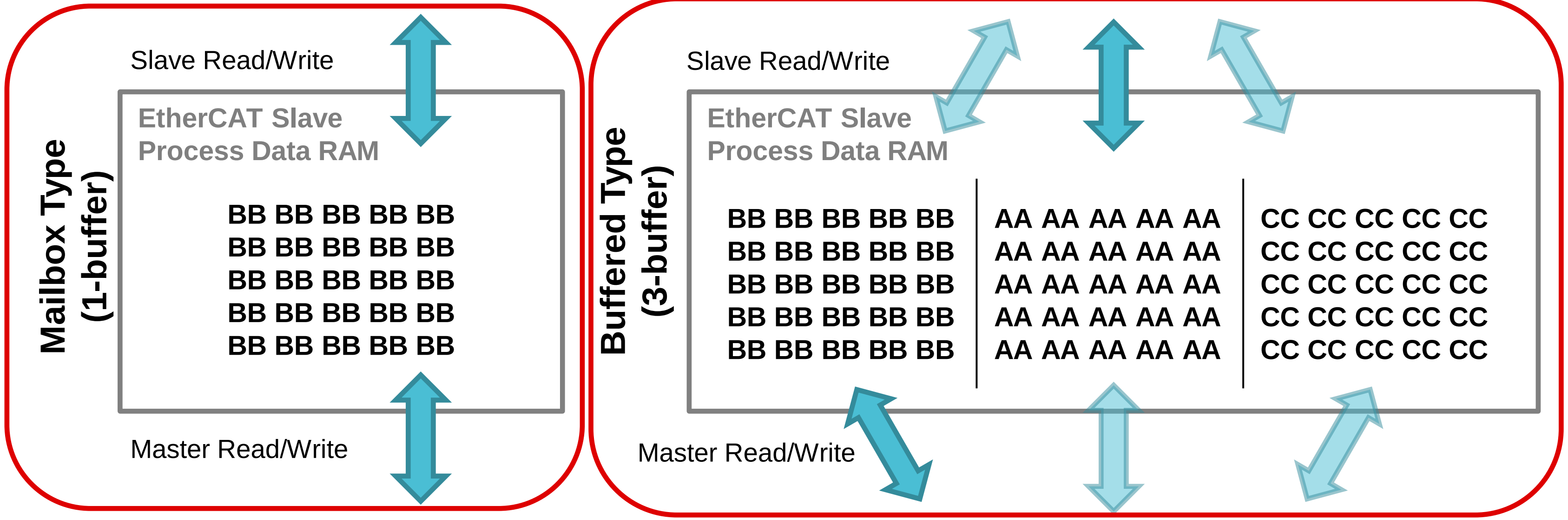
| | | | |
|---|---|---|---|
| EtherCAT Slave Controller Configuration Area | | | |
| VendorId | ProductCode | RevisionNo | SerialNo |
| Hardware Delays | | Bootstrap Mailbox Config | |
| Mailbox Sync Man Config | | Reserved | |
| Reserved | | | |
| Additional Information (Subdivided in Categories) … | | | |
| Category Strings | | | |
| Category Generals | | | |
| Category FMMU | | | |
| Category SyncManager | | | |
| Category Tx- / RxPDO for each PDO | | | |

# EtherCAT Slave – EPU: FMMU Details



- FMMUs map the ESC RAM interval into the global address space of the master and vice versa
- With FMMU, each slave reads and writes its data in the same position. Multiple slaves can share the same datagram
  – Without FMMU, each slave that needs to be addressed would require its own datagram

# EtherCAT Slave – EPU: SyncManager Details

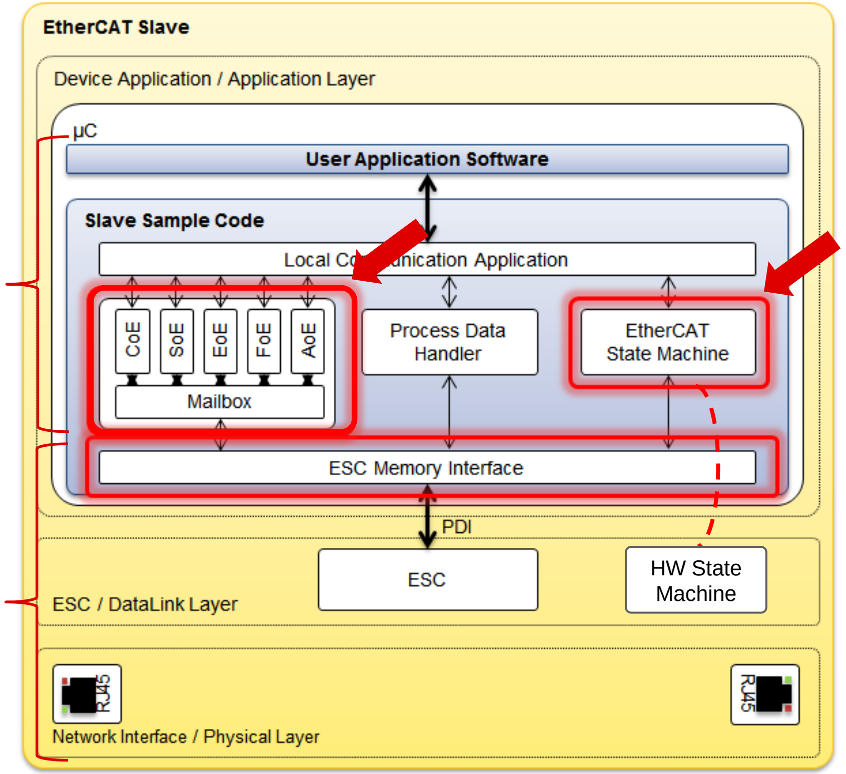


- SyncManagers protect the Process Data RAM interval from simultaneous access to maintain data consistency
- **SyncManager Mailbox (1-buffer) Type:** Used for non-process data communication
  - Writing side must write before reading side can read
  - Reading side must read before writing side can write again
- **SyncManager Buffered (3-buffer) Type:** Used for process data communication
  - 3 buffers guarantee consistent data delivery and access to the newest data any time

# EtherCAT Slave – Slave Stack Structure

Software

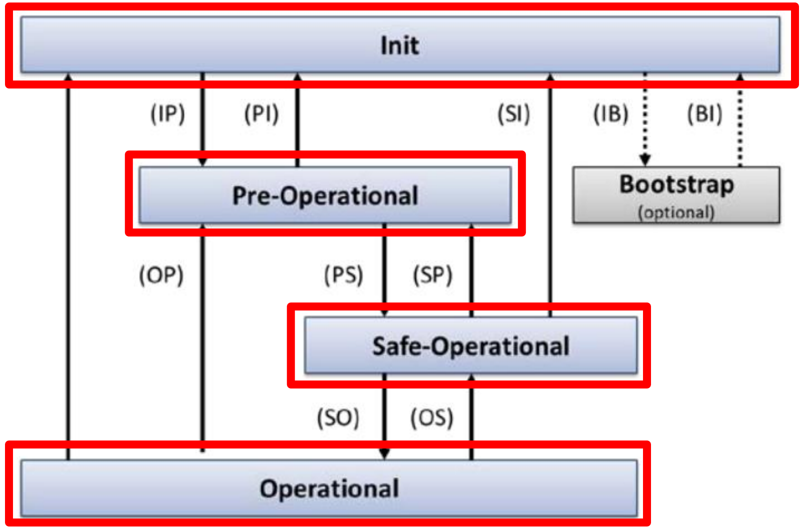
Hardware



# EtherCAT Slave – Communication Profiles Details

| Profiles | Details |
|---|---|
| Ethernet over EtherCAT (EoE) | • Tunnels standard Ethernet communication (ex: TCP/IP) over EtherCAT<br>• Allows the master to optimize Ethernet communication without affecting the process data exchange |
| CAN application protocol over EtherCAT (CoE) | • Access of a CANopen object dictionary<br>• Recommended protocol for service data access<br>• Easy migration path from CANopen devices to EtherCAT device |
| File Access over EtherCAT (FoE) | • Download and upload files (ex: firmware download)<br>• Similar to Trivial File Transfer Protocol, RFC 1350<br>• Lean stack implementation, suitable for bootstrap loaders |
| Servo Drive over EtherCAT (SoE) | • Access the Servo Profile Identifier<br>• Implements service channel<br>  • Read/write to several elements of an IDN<br>  • Support of procedure commands |

# EtherCAT Slave – State Machine



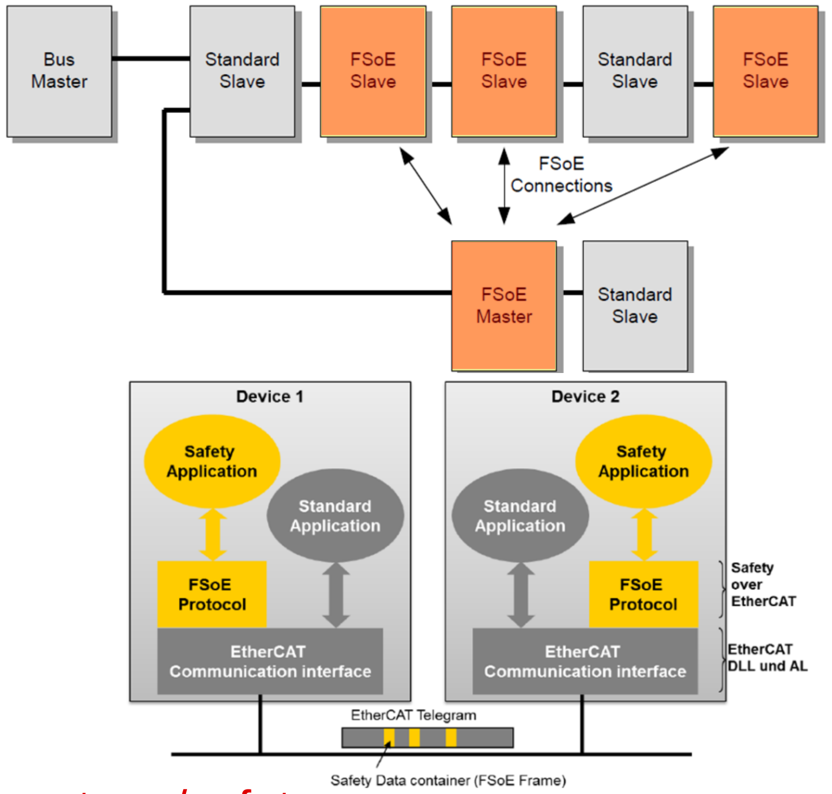
- **Init**
  - No communication on the application layer is available. The master has access only to the DL-information registers.
- **Pre-Op**
  - PREOP Pre-Operational state. Mailbox communication on the application layer available, but no process data communication available.
- **Safe-Op**
  - Safe-Operational state. Mailbox communication on the application layer, process (input) data communication available. In SafeOp only inputs are evaluated; outputs are kept in 'safe' state.
- **Operational**
  - Process data inputs and outputs are valid.

- Note that the Master *requests* each transition, but the slave must *confirm* it.

AL Control (0x0120) (in DPRAM Mem Space) AL Status (0x0130)

# EtherCAT Slave – Safety over EtherCAT (FSoE)

- EtherCAT safety support utilizes **FailSafe over EtherCAT** (FSoE) protocol
- Enables transmission of safety-related data in parallel with standard data on the same network
- Slaves in the network are identified as FSoE Master and FSoE supporting-slaves
- FSoE is designed to be used in conjunction with standard EtherCAT communication protocols



https://www.ethercat.org/safety

## More Information on EtherCAT

EtherCAT® Technology Group

www.ethercat.org