BECKHOFF

# EtherCAT Evaluation Kit Workshop

# Hardware & Software

# Handout

**Beckhoff Automation**
GmbH & Co. KG
Hülshorstweg 20
33415 Verl, Germany

Postfach 11 42
33398 Verl

Telefon: +49 5246 963 - 0
Vertrieb: +49 5246 963 - 1000
Service: +49 5246 963 - 460
Support: +49 5246 963 - 157

Fax Zentrale: - 149
Fax Vertrieb: - 198

www.beckhoff.de
info@beckhoff.de
vertrieb@beckhoff.de
service@beckhoff.de
support@beckhoff.de
international-sales@beckhoff.de
engineering@beckhoff.de

Geschäftsführer:
Dipl.-Phys. Hans Beckhoff

Registergericht:
Gütersloh HRA 7075

Ust.-Id.-Nr.: DE 815529334

Kreissparkasse Wiedenbrück
BIC: WELADED1WDB
IBAN: DE24 4785 3520 0000 0600 04 (EUR)
IBAN: DE02 4785 3520 0000 0600 12 (USD)
Deutsche Bank (EUR, USD)
BIC: DEUTDE3B489
IBAN: DE95 4807 0043 0373 6014 00

BECKHOFF

# EtherCAT Slave Development Workshop

1901

# Agenda

BECKHOFF

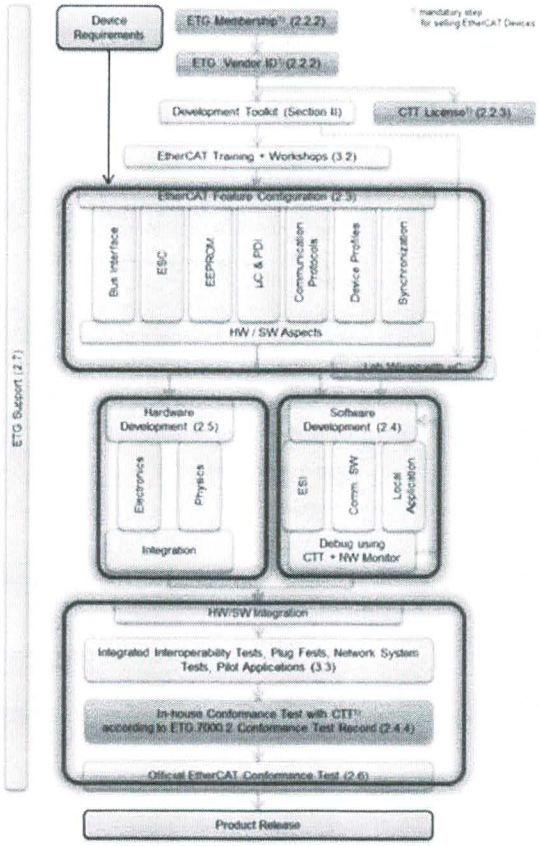
1. Slave Development Process overview
2. Design 2 EtherCAT slaves
   - Simple Digital IO slave
   - Standard Slave

Notes:

# Development process overview

BECKHOFF

1. Device Definition
2. Hardware Design
3. Software Development
4. Conformance Testing

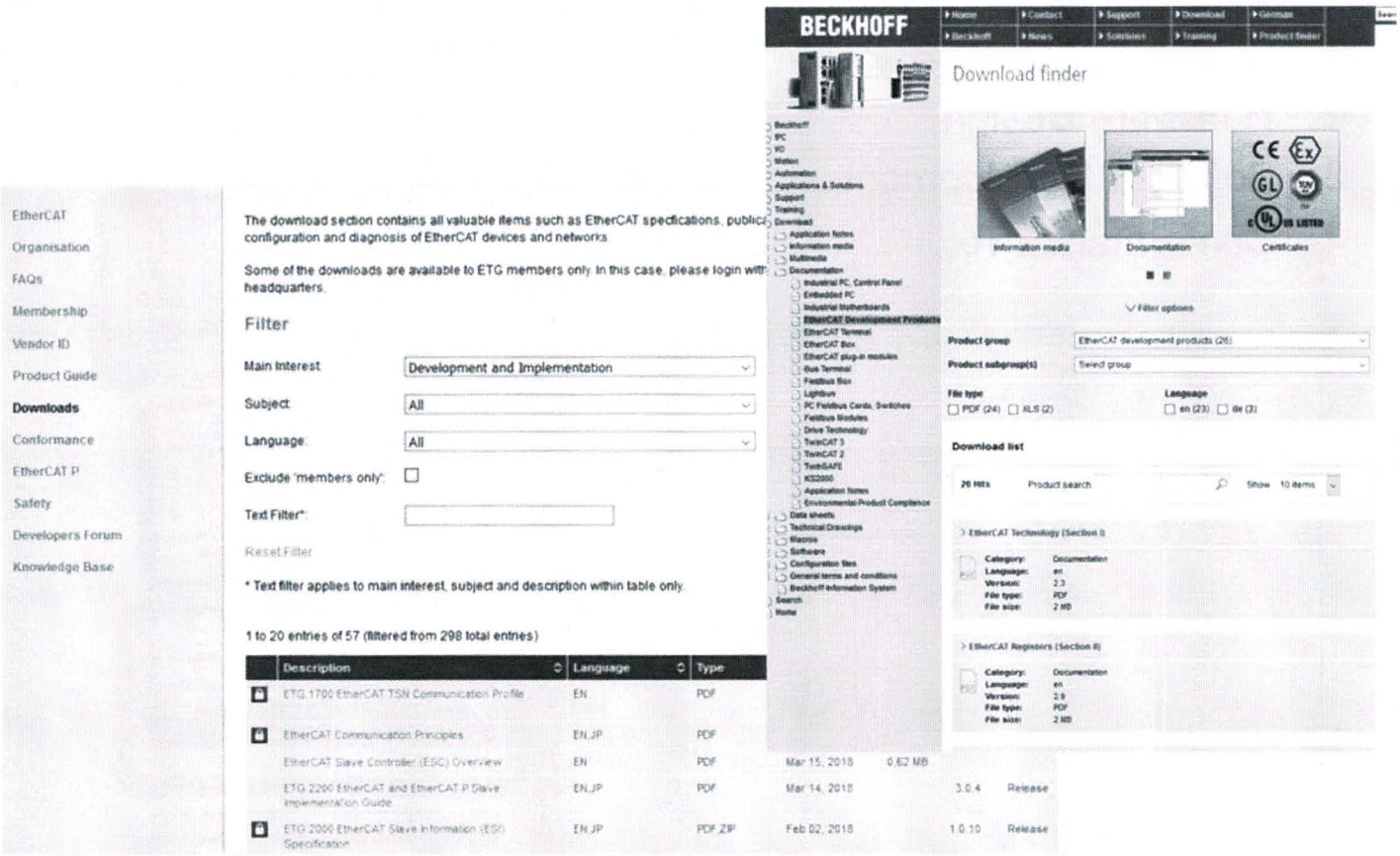



## Notes:

- Detailed Developement Process described in the Slave Implementation Guide: www.ethercat.org/ETG2200

# Development Documents

BECKHOFF

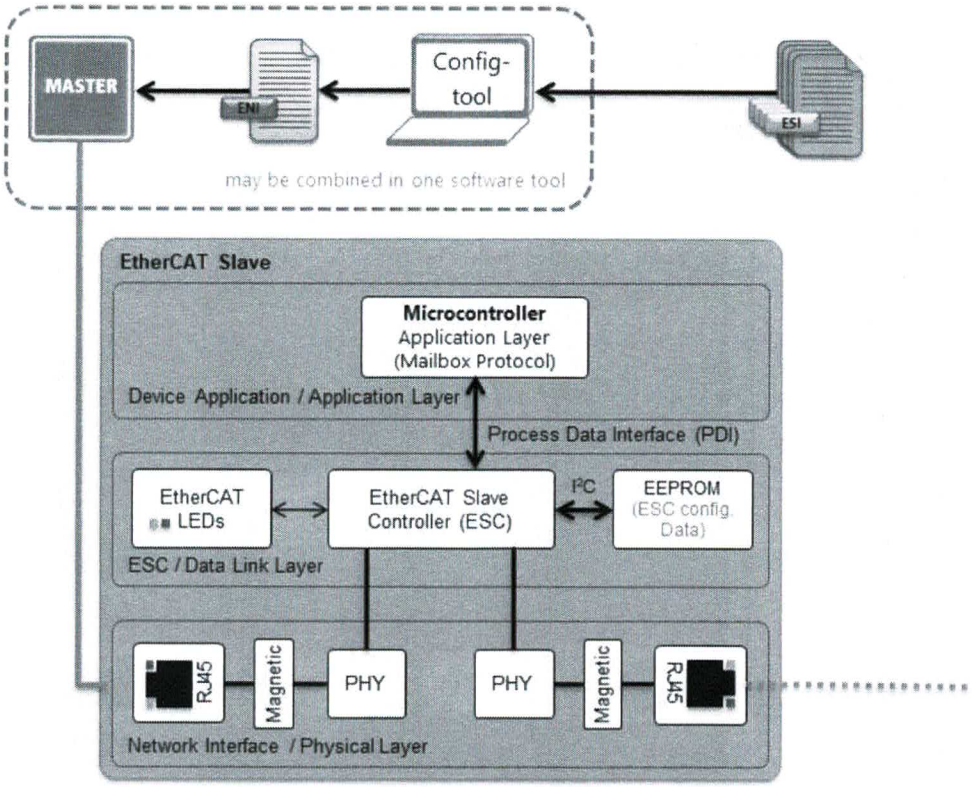



## Notes:

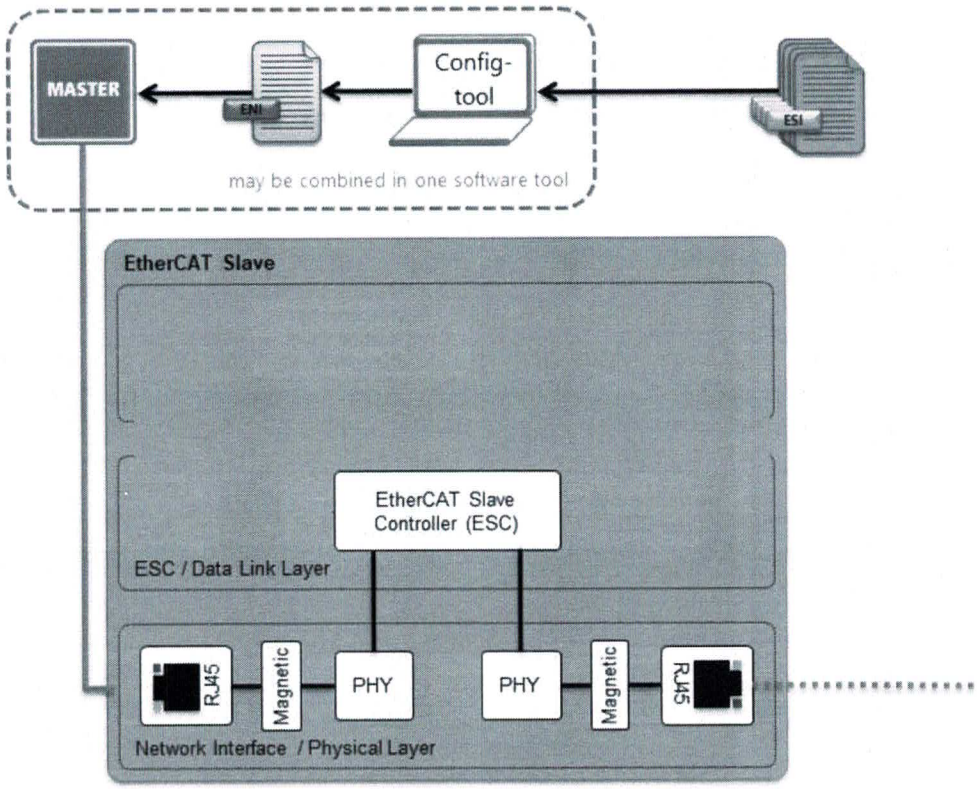
- https://www.ethercat.org/download
- www.beckhoff.de/download/ethercat-development-products/

# Slave Structure

BECKHOFF




Notes:




## Notes:

- A slave may have up to 4 Ports

- 100 BASE-TX
  - Twisted pair, shielded
  - PHY and magnetics needed
  - Up to 100m between two EtherCAT slaves

- 100 BASE-FX
  - PHY with Fiber optic transceiver
  - Up to 20km between two EtherCAT slaves

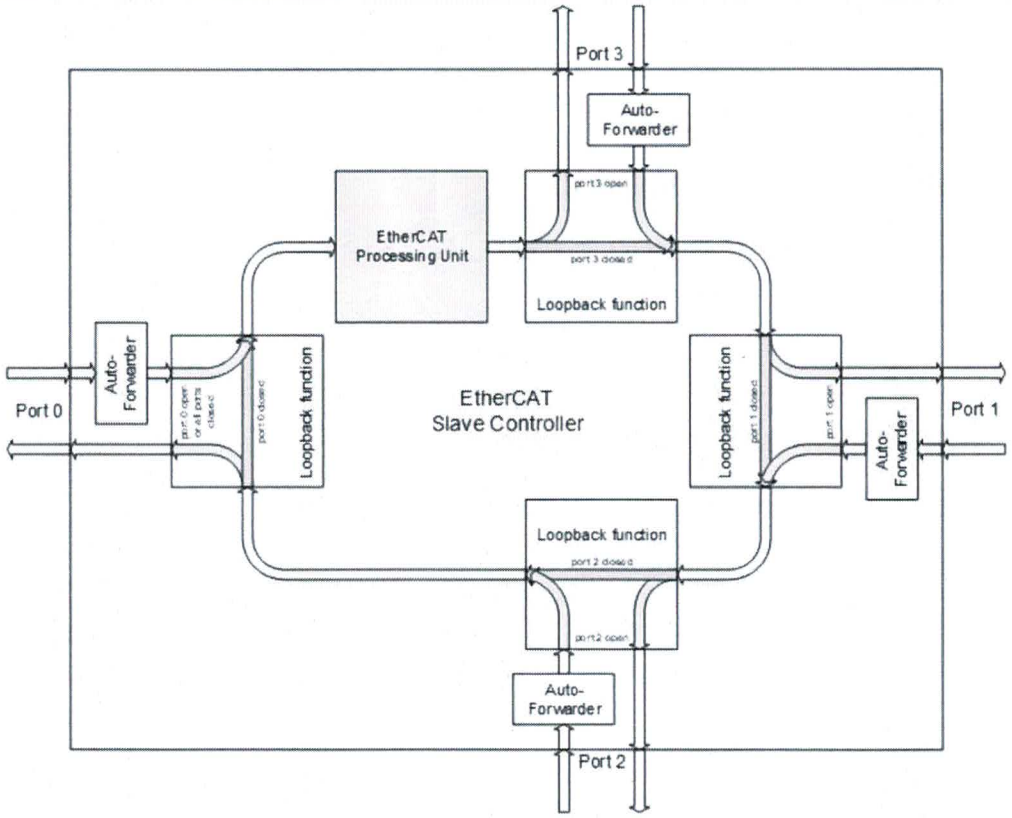
Notes:

- Up to 4 EtherCAT ports are possible on one ESC (with reduced PDI)

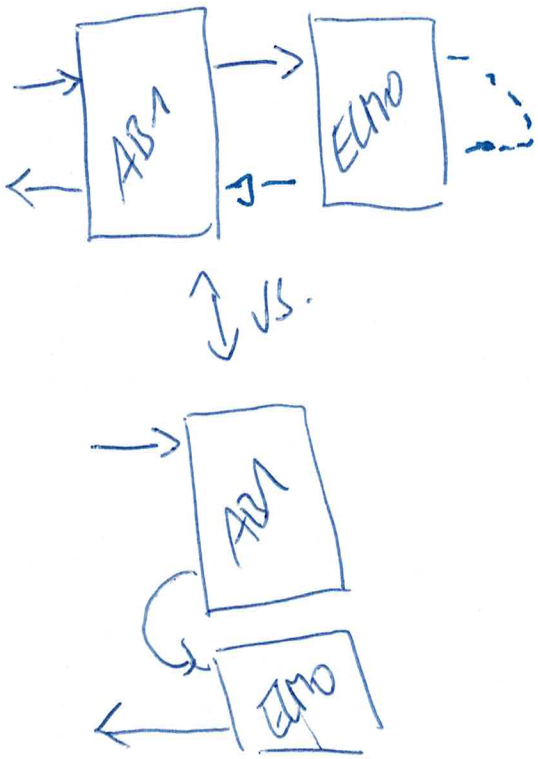
- PHY Selection guide (www.beckhoff.com/download/ethercat-development-products/)
  - Requirements to PHYs
  - Examples for PHYs

# ESC Ports | Frame Processing Order

BECKHOFF

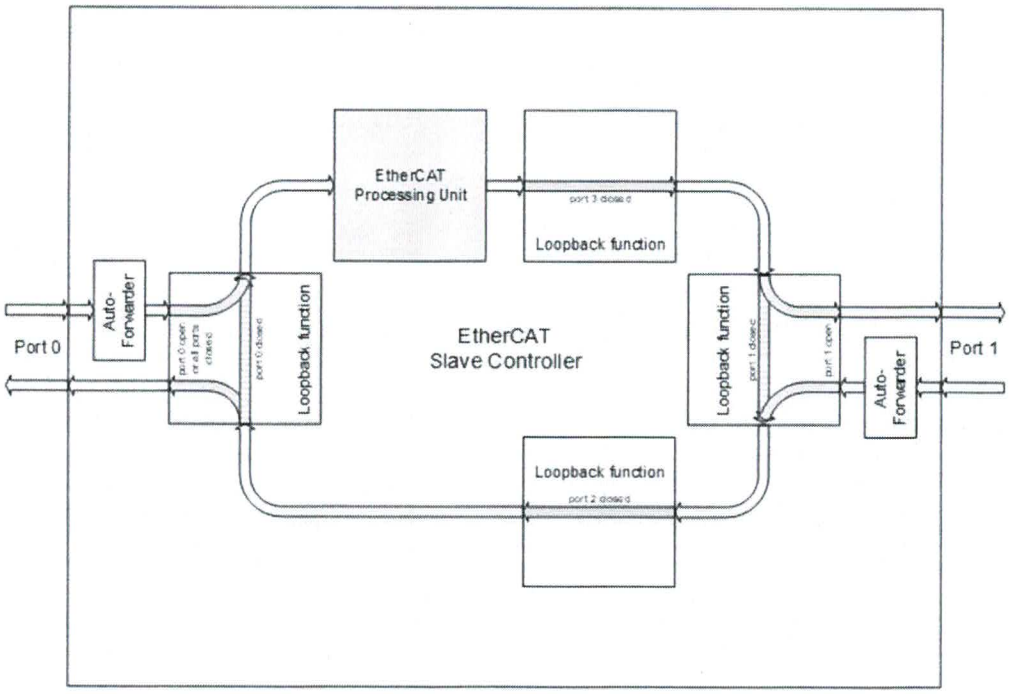



## Notes:

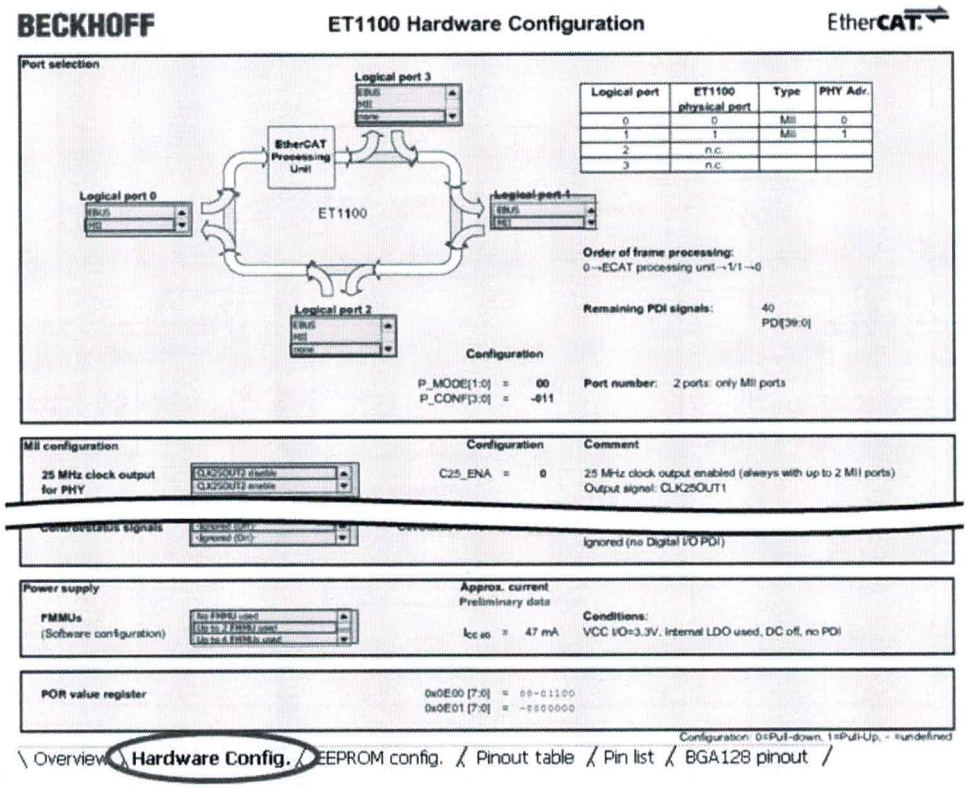


# ESC Ports | Frame Processing Order

BECKHOFF



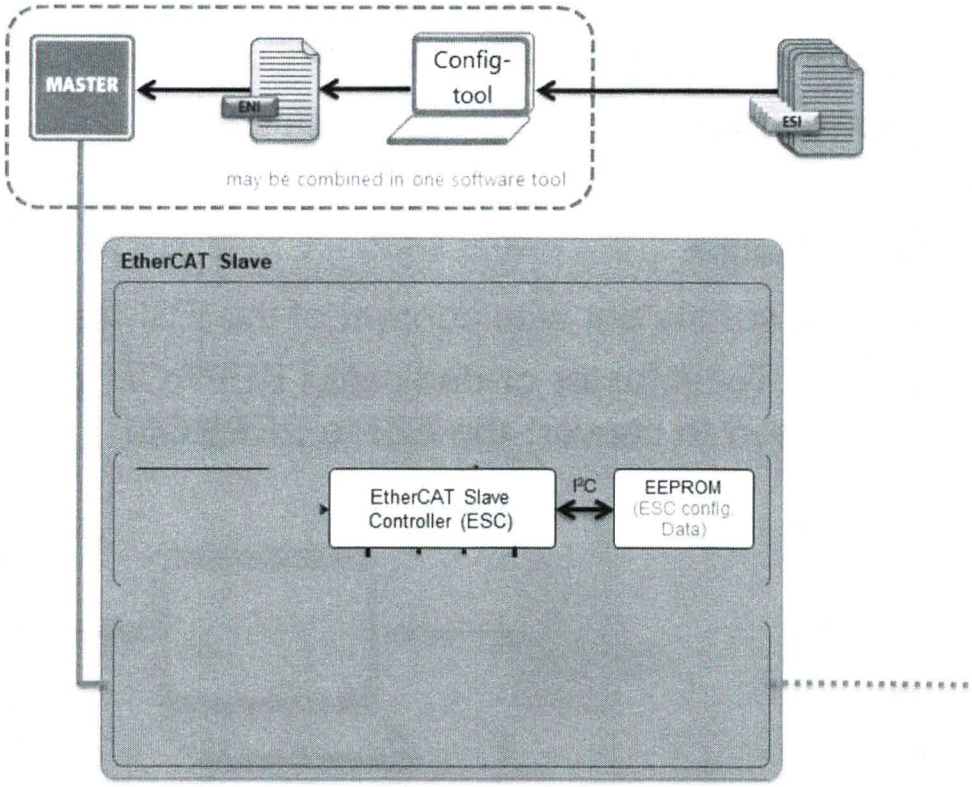

Notes:




## Notes:

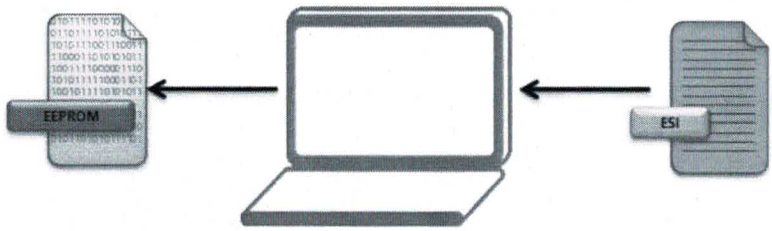
Use ET11xx_Configuration_and_Pinout.xls (www.beckhoff.com/download/ethercat-development-products/)→ ET1100 /1200 Configuration and Pinout

Connection usually ALWAYS via 100 BASE-TX or FX
↳ direct interface not allow VendorA/VendorB


MASTER
ENI
Config-
tool
ESI
may be combined in one software tool
EtherCAT Slave
EtherCAT Slave
Controller (ESC)
I²C
EEPROM
(ESC config.
Data)

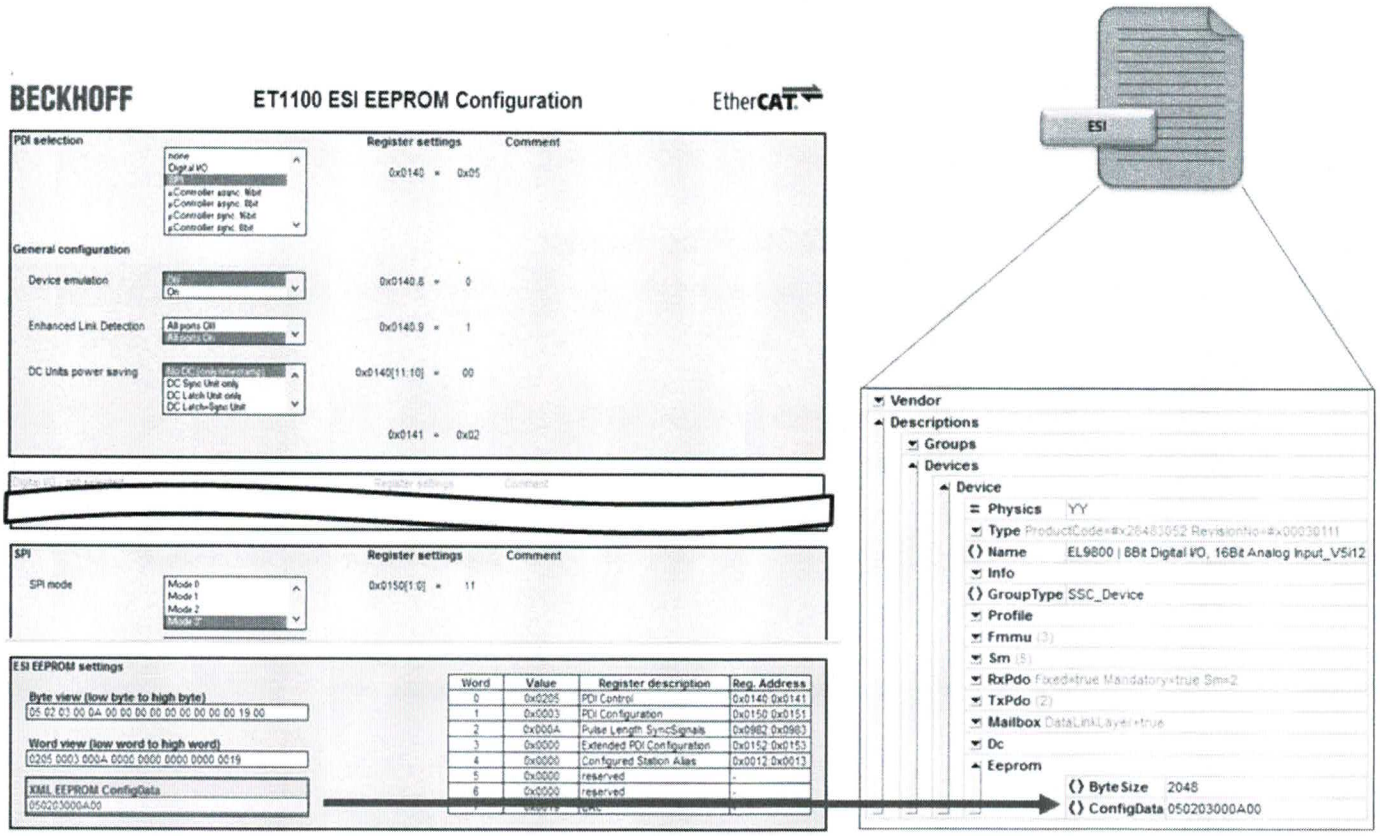
Notes:

- ESC/PDI configuration parameters
- Identification data (Vendor ID, Product Code, Revision Number)
- Basic Information to run the slave application
  - Process data information
  - Synchronisation
- Data shall be consistent with content of the ESI file
- Configuration Tool/Master or dedicated EEPROM programming tool can be used to convert the ESI to EEPROM



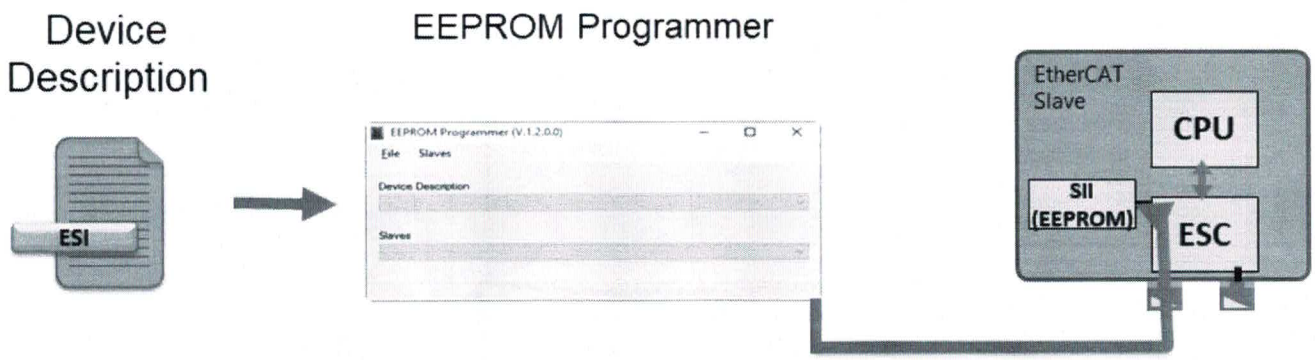

## Notes:

- ESC: EtherCAT Slave Controller
- ESI: Slave Information Interface (Description file for the EtherCAT Slave
- PDI : Process Data Interface (connection between the ESC and the Slave Processor or digital IO data)




## Notes:

- Use ET11xx_Configuration_and_Pinout.xls (www.beckhoff.com/download/ethercat-development-products/)→ ET1100 /1200 Configuration and Pinout
- ESC Configuration Data : /EtherCATInfo/Descriptions/Devices/Device/Eeprom/ConfigData

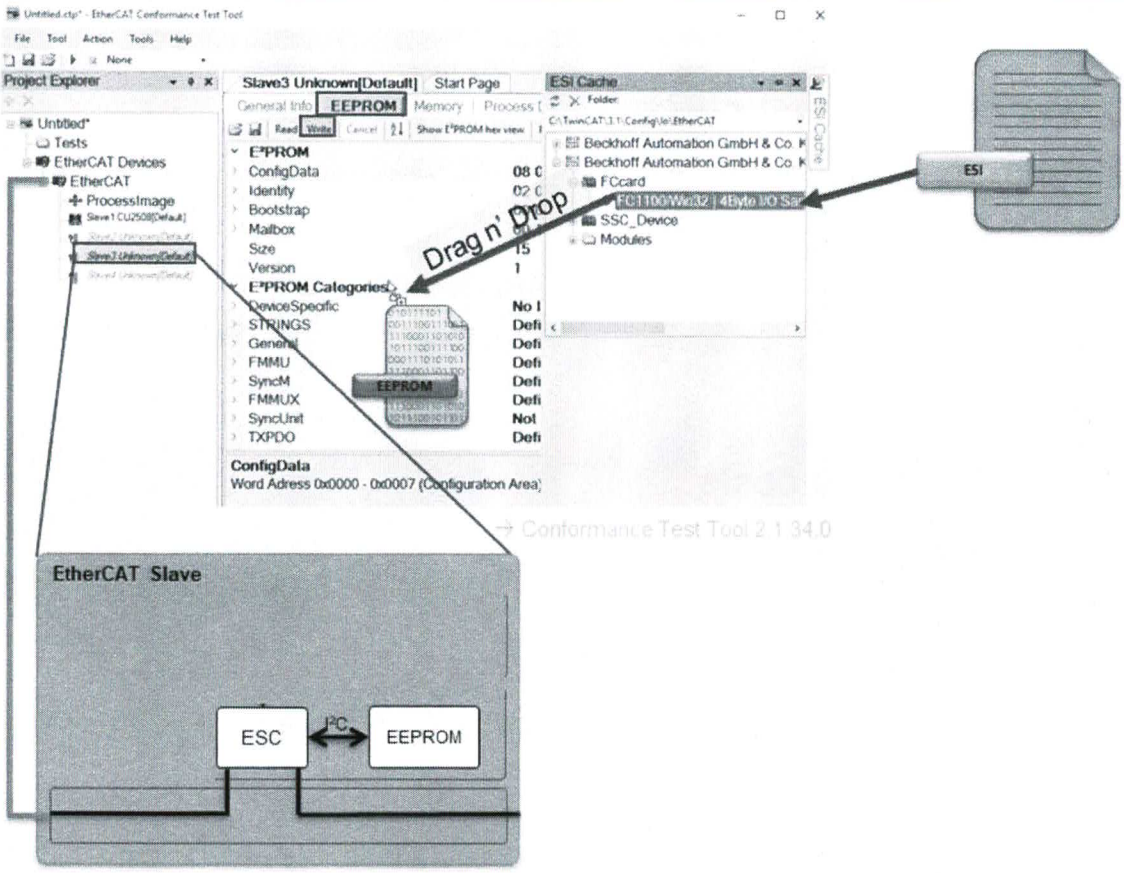



## Notes:

- Programm the EEPROM

# ESC EEPROM | Programming (with CTT)

BECKHOFF

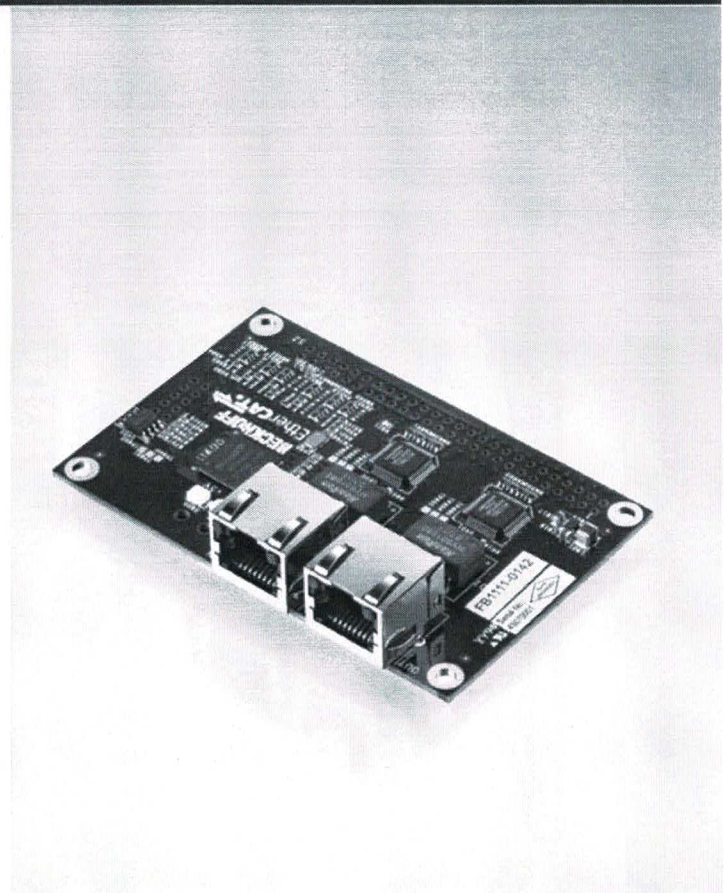



Notes:

# New Design Alternative | Communication board

BECKHOFF

- PRO:
  - Lower development costs
  - Time to market
  - Less network know-how required
- CONTRA:
  - Higher hardware costs
  - Form factor restrictions



Notes:

ET1100
~ 7-10€ / Piece

## Slave 1 | Device Requirements

BECKHOFF

- 16 Bit digital input
- 16 Bit digital output
- Further slave shall be connectable
- No parameter/configuration required

=> Simple EtherCAT Slave

Notes:

## Device Requirements => EtherCAT features

BECKHOFF

- Digital IO process data and no parameters required
  - ⇒No processor needed
  - ⇒Handled by the Digital IO PDI of the ESC
- 16 Bit digital input / 16 Bit digital output
  - ⇒3 Sync Manager
  - ⇒2 FMMUs
- Further slave shall be connectable
  - ⇒2 MII Ports

Notes:

- SyncManager : 1 for Inputs and 1 for each output byte
- FMMUs : 1 for each process data direction
- PDI : Process Data Interface
- FMMU : Fieldbus Memory Management Unit

# EtherCAT features

BECKHOFF

- 32Bit Digital IO PDI
- 3 Sync Manager
- 2 FMMUs
- MII Ports
- Low Cost

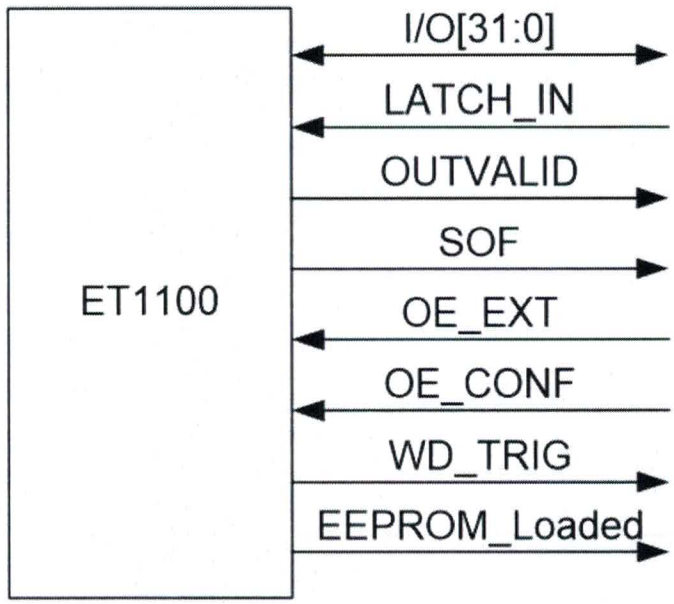
| Name | ET1100 | ET18xx |
|---|---|---|
| Type | ASIC ✓ | FPGA/Configurable IP Core ✗ |
| uController Interface | serial/parallel (8\|16bit)* | serial/parallel (8\|16bit) AVALON®/OPB®/PLB®* |
| Digital I/O | 8-32* ✓ | 8-32* ✓ |
| DPRAM | 8 kByte | 1...60 kByte* |
| SyncManager | 8 ✓ | 0...8* ✓ |
| FMMU | 8 ✓ | 0...8* ✓ |
| DC Support | yes | yes* |
| No. of Ports | 2-4(MII/EBUS) ✓ | 1-3(MII/RMII/RGMII only 2) ✓ |

*configurable

Notes:

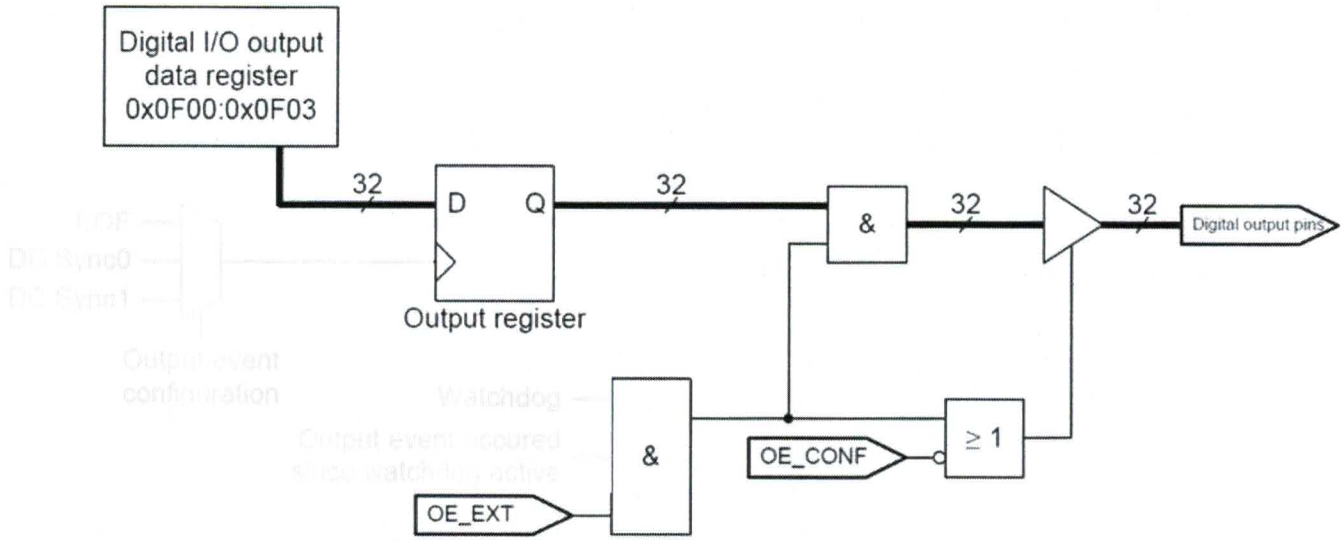
- Up to 32 I/Os
- 3,3V or 5V signals
- -3mA..4mA

↑ it can sink current.




## Notes:

- 64 I/Os in multiplexed mode with ET1100 is possible
- 3,3V or 5V depends on Vcc (Vcc 3,3V => IO 3,3 V)
- Signals:
  - I/O[31:0] - Input/Output or Bidirectional data
  - LATCH_IN - External data latch signal
  - OUTVALID - Output data is valid/Output event
  - SOF OUT - Start of Frame
  - OE_EXT - Output Enable
  - OE_CONF - Output Enable Configuration
  - WD_TRIG - Watchdog Trigger
  - EEPROM_LOADED - PDI is active, EEPROM is loaded

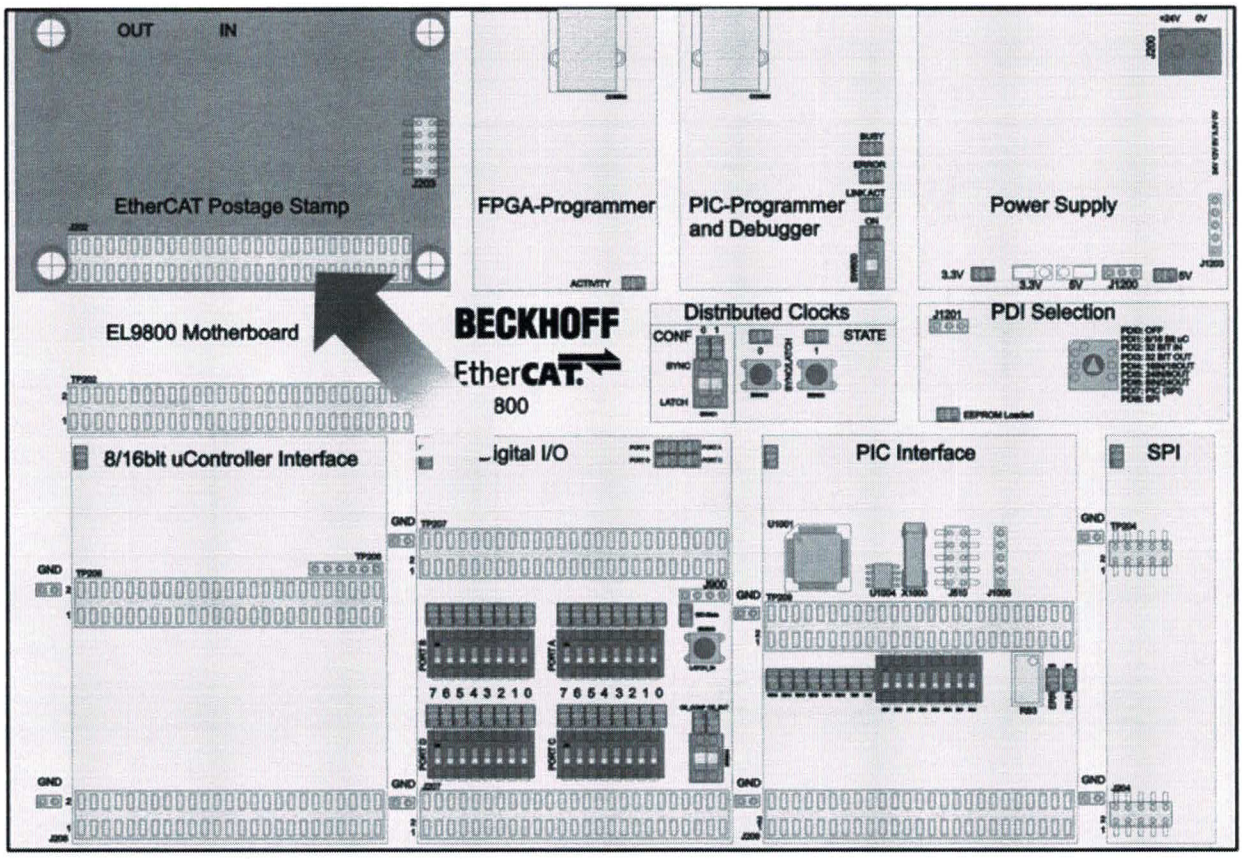



## Notes:

- Schema taken from the ET1100 Datasheet (www.beckhoff.com/download/ethercat-development-products/)
- OE_EXT : Output enable
- OE_CONF : Output enable configuration

# Realize with EL9820

BECKHOFF

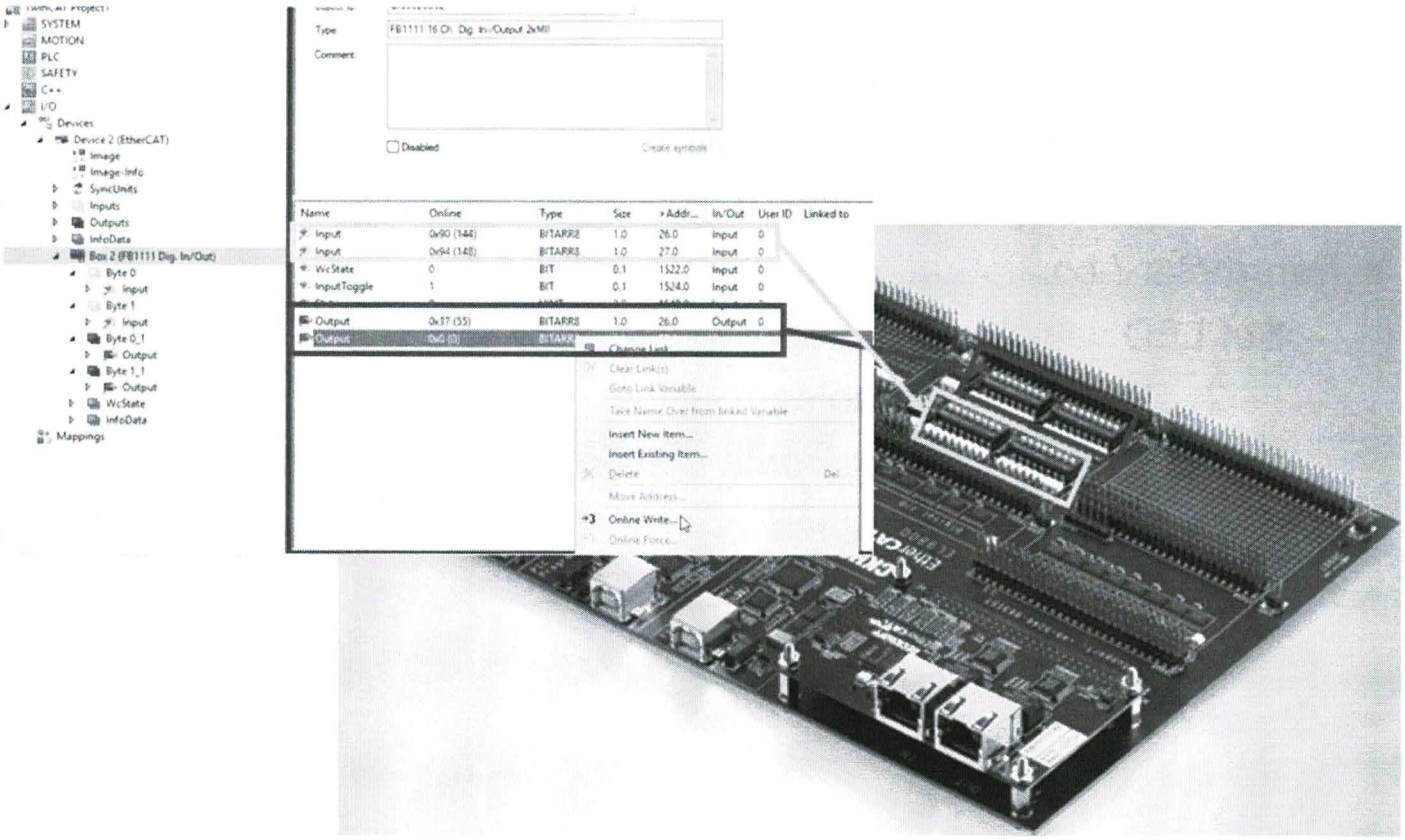



## Notes:

- Power off EL9800
- Set PDI Selector to 0 ("OFF")
- Power on EL9800
- Program "FB1111 16 Ch. Dig. In-/Output 2xMII" to EEPROM
  - ESI file "Beckhoff FB1XXX.xml" needs to be located in "c:\TwinCAT\3.1\Config\io\EtherCAT\"
- Power off EL9800
- Set PDI Selector to 4 ("32 Bit digital Interface - 16 Inputs / 16 Outputs")
- Power on EL9800

# Process data visualization

BECKHOFF




## Notes:

- Virtual Input Data:
  - WcState : Indicates if the process datagram was handled correct (Working counter is valid)
  - InputToggle : toggles between 0 and 1 on each valid received process datagram
  - State : Indicates the state of the slave (EtherCAT state, Link state, etc.)
- Write Output Data :select the Output data, open the context menu and select “Online Write”

- Process Data (cyclic)
  - Addition/Subtraction of two 16Bit Values
    - 2x 16Bit Output variables (master to slave)
    - 1x 16Bit Input variables (slave to master)
- Parameter / configuration required (acyclic)
  - Set LED values
    - 2x BOOLEAN variables
  - Select between process data addition and subtraction
    - 1x BOOLEAN variables
- Further slave shall be connectable

=> Standard Slave

Notes:

## Device Requirements => EtherCAT/Hardware features BECKHOFF

- None digital IO process data (and parameters required)
  ⇒Processor needed
  ⇒Slave application needed
- Process data and mailbox data
  ⇒4 Sync Manager
  ⇒3 FMMUs
- At least 274Byte RAM (in the ESC) required
- Further slave shall be connectable
  ⇒2 MII Ports

- Processor requirements (slave application)
  ⇒~30KB Program/Data Memory / ~15KB RAM

Notes:

- 4 Sync Manager
  - Mailbox Output (master to slave)
  - Mailbox Input (slave to master
  - Process data output (master to slave)
  - Process data input (slave to master)

- 3 FMMU
  - Process data output
  - Process data input
  - Mailbox state (indicates if the slave has written new mailbox data)

- Required Process data RAM caluclation
  - Acyclic communication 128 Bytes (default value) per direction (handled with 1 buffer SyncManager)
  - Cyclic Output 2x 2Byte (handled with 3 buffer SyncManager)
  - Cyclic Input 1x 2Byte (handled with 3 buffer SyncManager)

  => 128 + 128 + 2*2*3 + 2*3 = 274Byte RAM required

# EtherCAT features

BECKHOFF

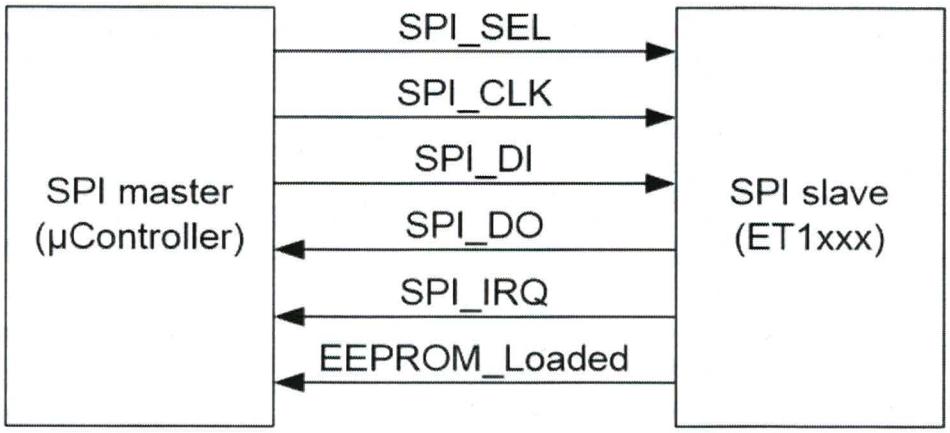
- SPI PDI
- 4 Sync Manager
- 274Byte DPRAM
- 3 FMMUs
- MII Ports
- Low Cost

| Name | ET1100 | ET18xx |
|---|---|---|
| Type | ASIC ✓ | FPGA/Configurable IP Core ✗ |
| uController Interface | serial/parallel (8\|16bit)* ✓ | serial/parallel (8\|16bit) AVALON®/OPB®/PLB®* ✓ |
| Digital I/O | 8-32* | 8-32* |
| DPRAM | 8 kByte ✓ | 1...60 kByte* ✓ |
| SyncManager | 8 ✓ | 0...8* ✓ |
| FMMU | 8 ✓ | 0...8* ✓ |
| DC Support | yes | yes* |
| No. of Ports | 2-4(MII/EBUS)* ✓ | 1-3(MII/RMII/RGMII only 2) ✓ |

*configurable

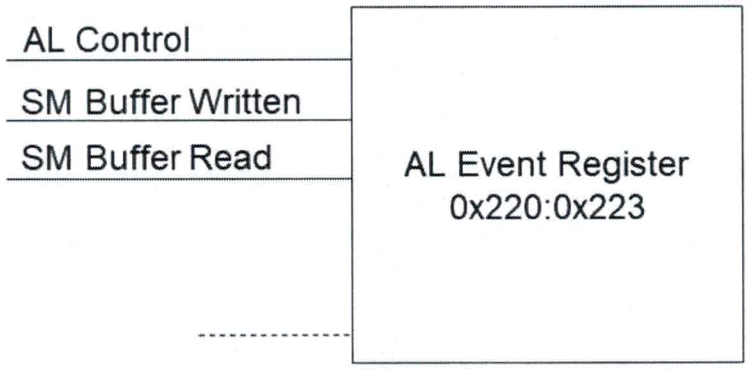
Notes:

- Up to 20 MHz
- µC is SPI master
- Little endian format
- Address will be incremented automatically (sequential read)




## Notes:

- EEPROM Loaded shall be monitored by the slave application after power on
- If the EEPROM Loaded pin is high the PDI is available and the slave application can access the ESC



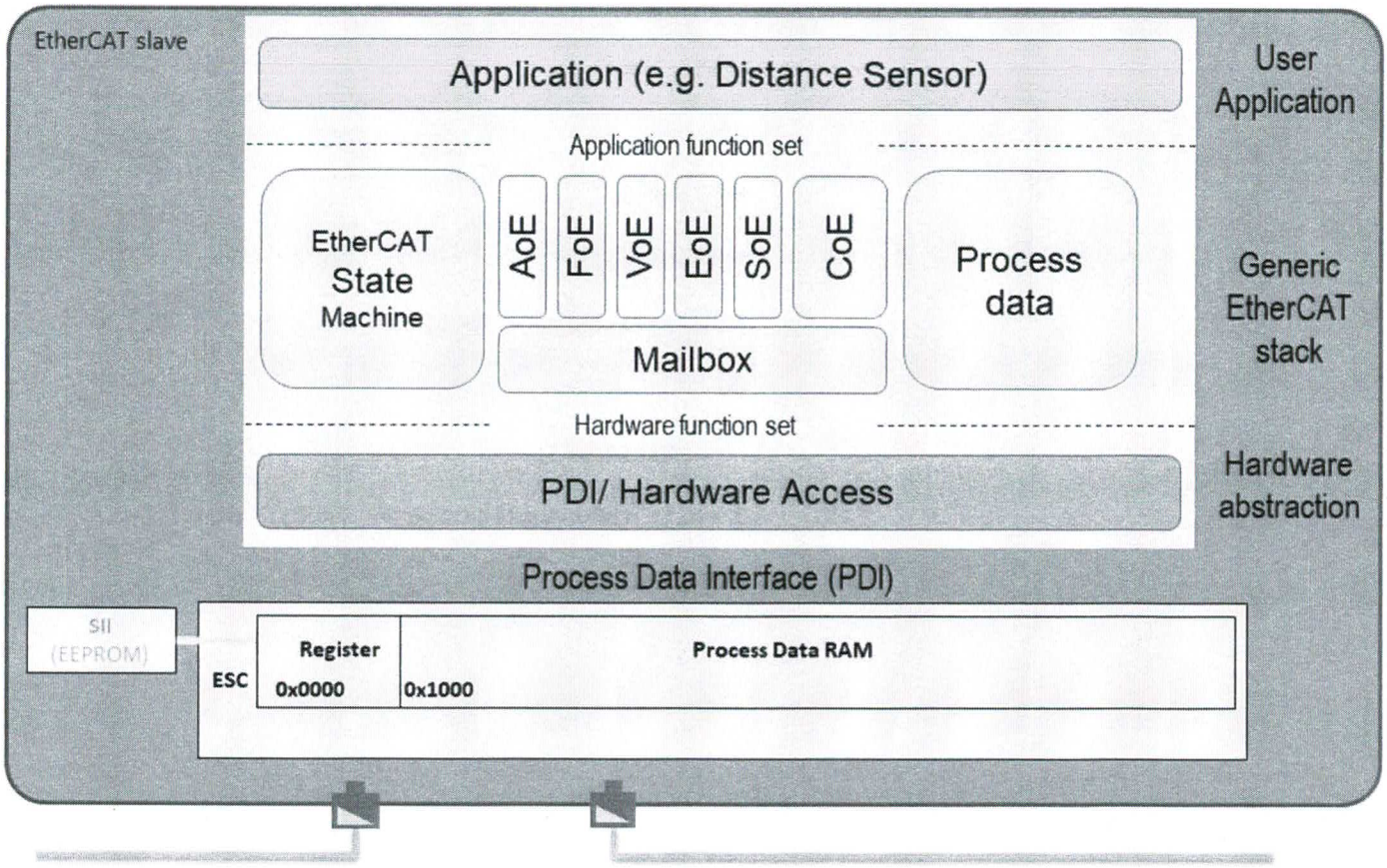

| Event Trigger | Description | Event Acknowledge | Register |
|---|---|---|---|
| AL Control Event | New state request (Master has written to register 0x120) | Reading register 0x120 via PDI | 0x220.0 |
| SM Write Event | Master has written SM memory | Reading memory via PDI | 0x220.n (n=23:8) |
| SM Read Event | Master has read SM memory | Write memory via PDI | |

## Notes:

SM : Sync Manager
PDI : Process data interface

# Slave Application Structure

BECKHOFF



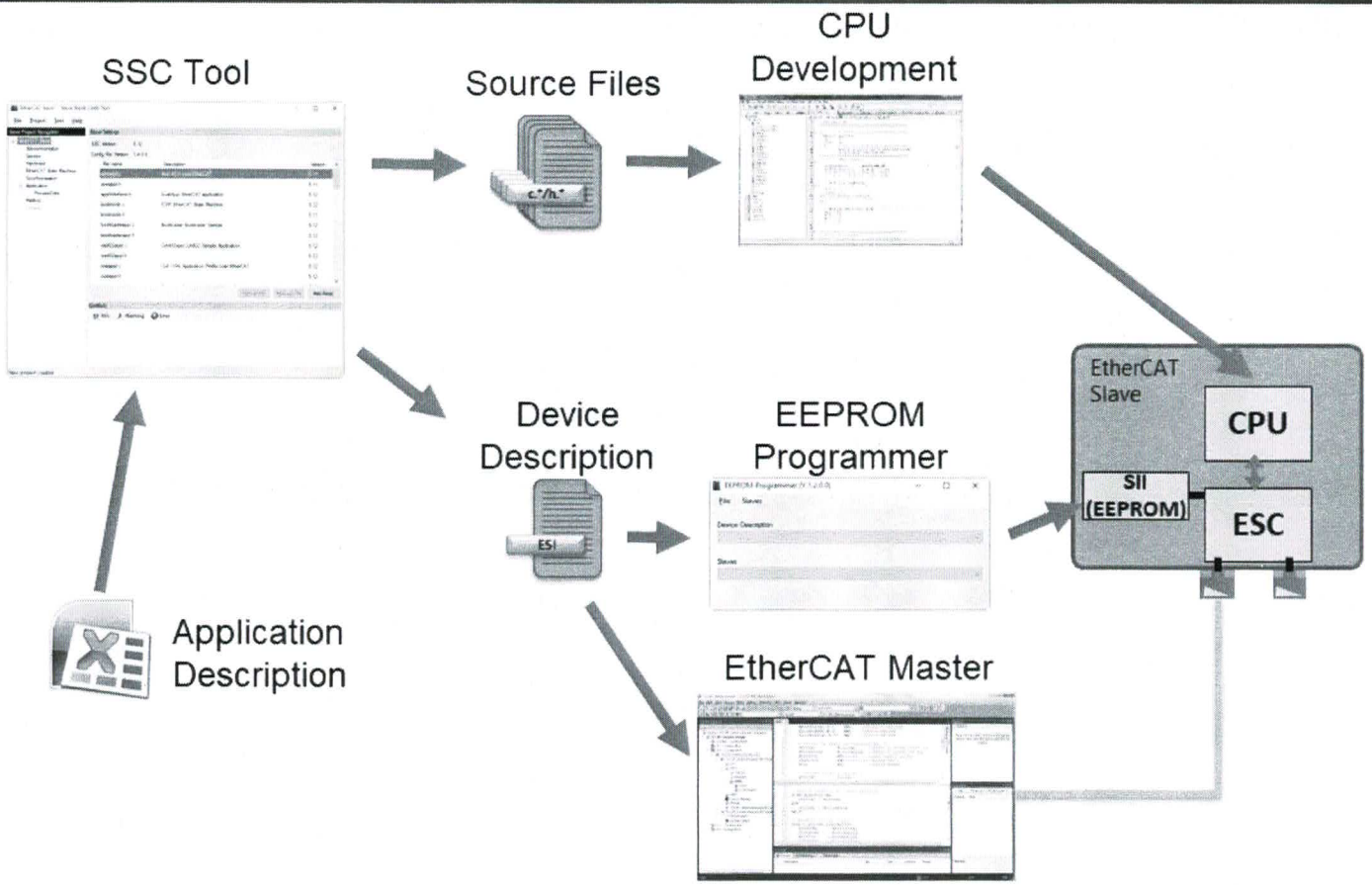

Notes:

```
void main(void)
{
    HW_Init();
    MainInit();
    bRunApplication = TRUE;
    do
    {
        MainLoop();
    } while (bRunApplication == TRUE);
    HW_Release();
}
```

Notes:
- Initialization
- MainLoop:
  - State Machine
  - Mailbox handling
  - Process data handling (if no interrupts configured)

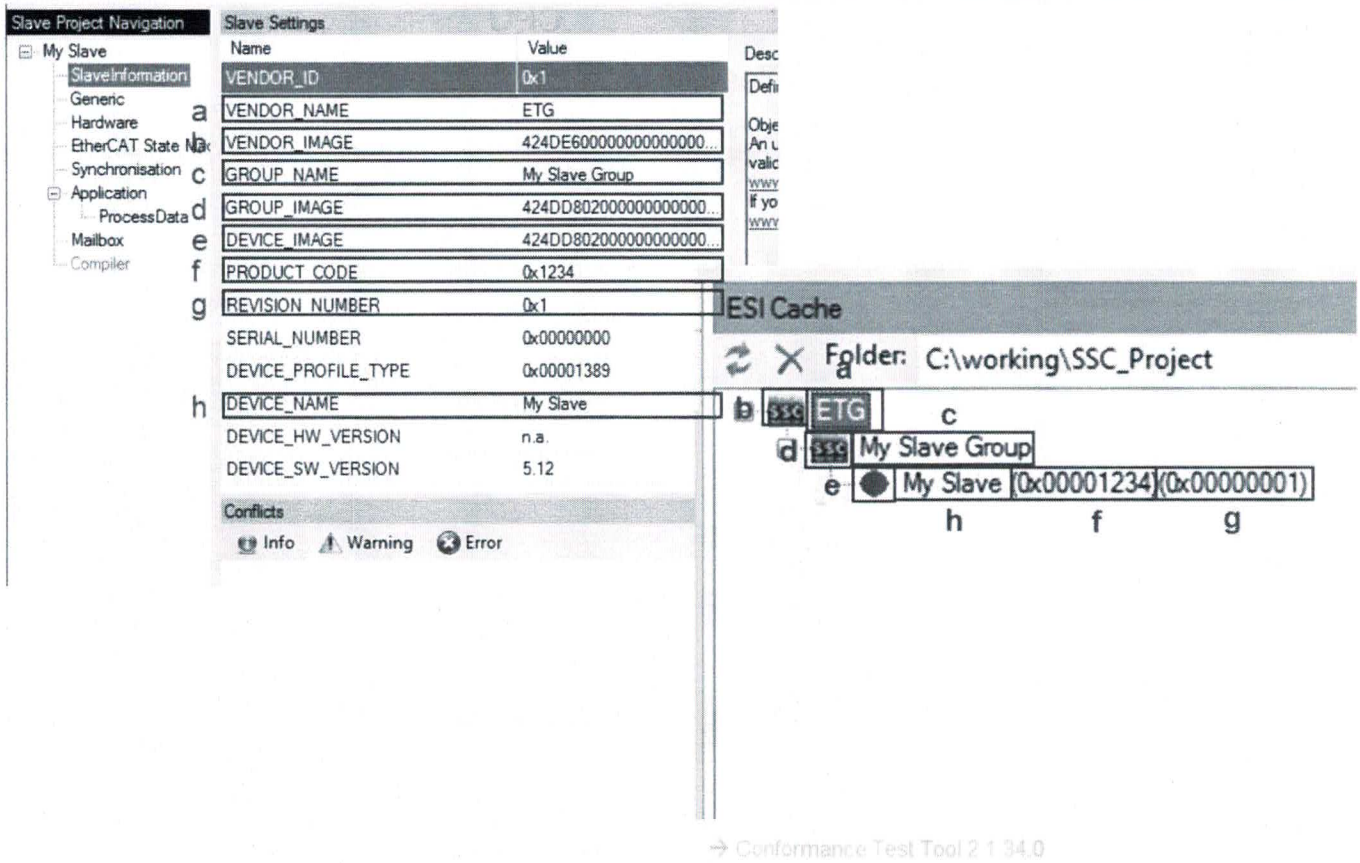
# SSC Tool | Progress

BECKHOFF




## Notes:

- Application Import is optional
  - Microsoft Excel is required


Slave Project Navigation
My Slave
SlaveInformation
Generic
Hardware
EtherCAT State Ma
Synchronisation
Application
ProcessData
Mailbox
Compiler
Slave Settings
Name
Value
VENDOR_ID
0x1
VENDOR_NAME
ETG
VENDOR_IMAGE
424DE600000000000000...
GROUP_NAME
My Slave Group
GROUP_IMAGE
424DD802000000000000...
DEVICE_IMAGE
424DD802000000000000...
PRODUCT_CODE
0x1234
REVISION_NUMBER
0x1
SERIAL_NUMBER
0x00000000
DEVICE_PROFILE_TYPE
0x00001389
DEVICE_NAME
My Slave
DEVICE_HW_VERSION
n.a.
DEVICE_SW_VERSION
5.12
Conflicts
Info
Warning
Error
ESI Cache
Folder:
C:\working\SSC_Project
ETG
My Slave Group
My Slave
(0x00001234)
(0x00000001)
a
b
c
d
e
f
g
h

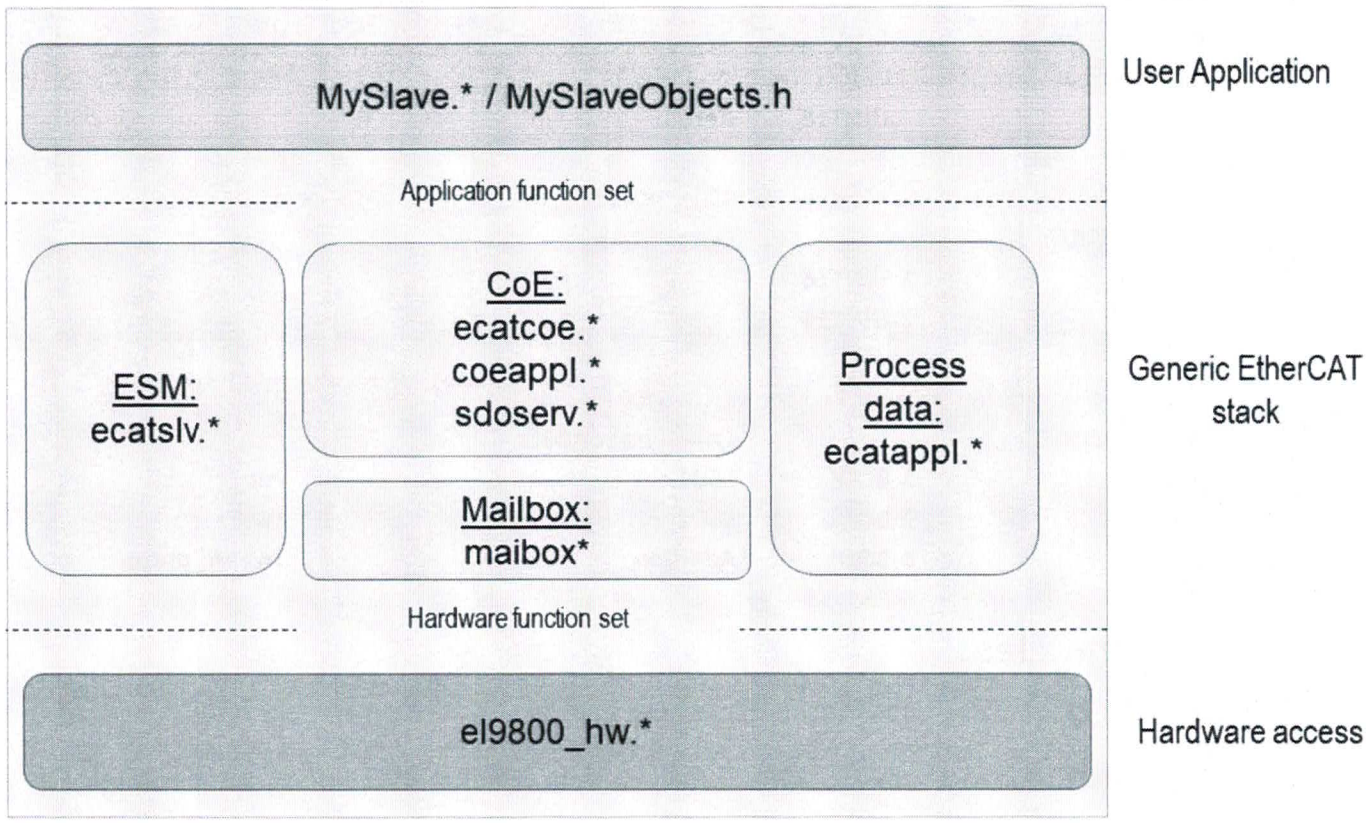
Notes:

| Index | ObjectCode | SI | DataType | Name | Default | B/S | Access |
|---|---|---|---|---|---|---|---|
| //0x6nnx | Input Data of the Module (0x6000 - 0x6FFF) | | | | | | |
| 0x6000 | | | UINT16 | Result | | | |
| | | | | | | | |
| //0x7nnx | Output Data of the Module (0x7000 - 0x7FFF) | | | | | | |
| 0x7000 | ARRAY | | | OutputData | | | |
| | | 1 | UINT16 | | | | |
| | | 2 | UINT16 | | | | |
| | | | | | | | |
| //0x8nnx | Configuration Data of the Module (0x8000 - 0x8FFF) | | | | | | |
| 0x8000 | RECORD | | | ConfigurationData | | | |
| | | 1 | BOOL | LED1 | | | rw |
| | | 2 | BOOL | LED2 | | | rw |
| | | 3 | BOOL | Addition | | | ro , wr_preop |
| | | 4 | Pad_5 | | | | |

- Generate the files:
  - MySlave.c
  - MySlave.h
  - MySlaveObjects.h

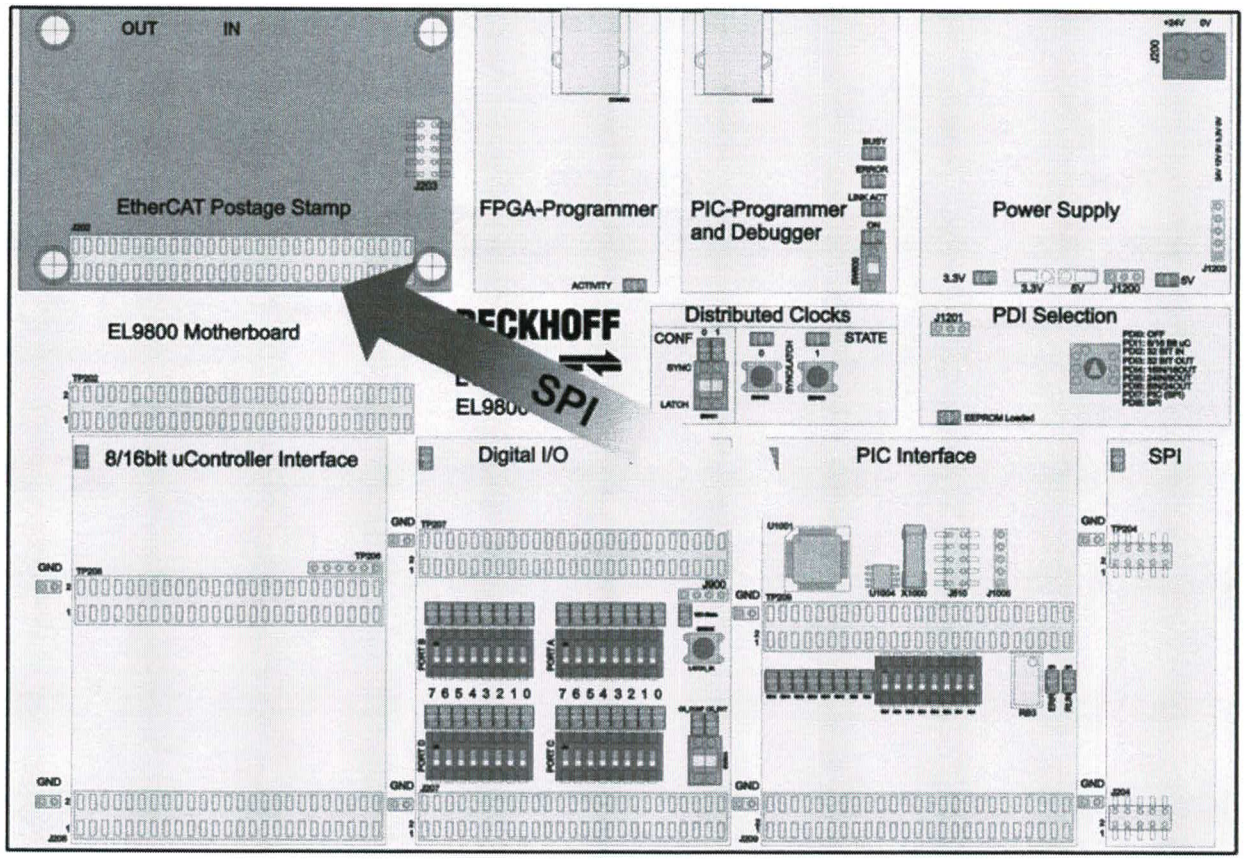
## Notes:

- Open a new Application definition table (Tool-> Application-> Create New)
  - The full syntax of the table is defined in the Slave Stack Code Application Note (download : www.beckhoff.com/download/ethercat-development-products/)
- MySlave.c/.h
  - Includes all application related functions and definitions
- MySlaveObjects.h
  - Includes the structures and variables defined in the spreadsheet



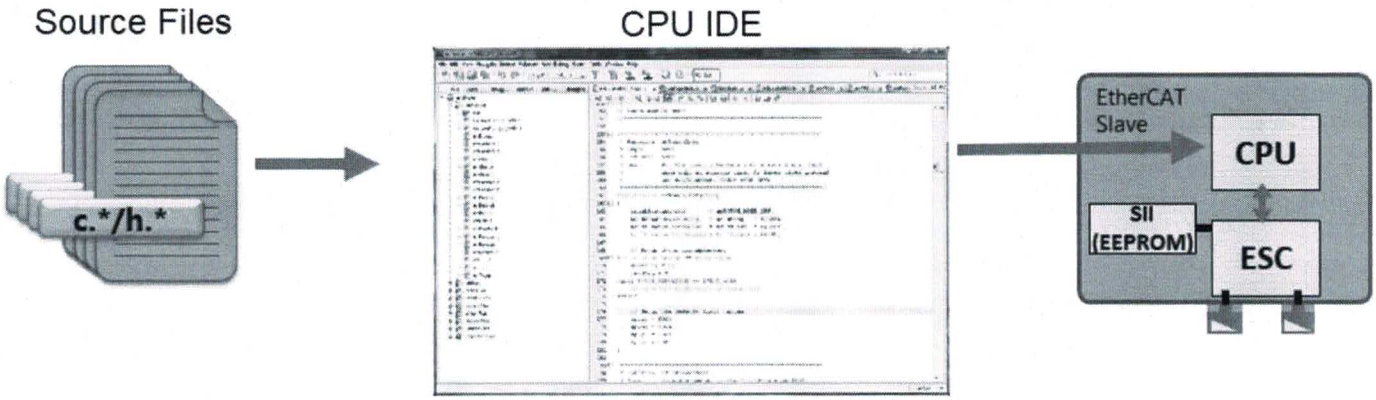

## Notes:

- Disable “DC_SUPPORTED” and “AL_EVENT_ENABLED” (for initial testing, can be enabled afterwards)
- In addition to the listed slave files the following files are created
  - My Slave.esp : Contains the Slave project including the slave files and all settings
  - My Slave.xlsx : Application spreadsheet
  - My Slave.xml : ESI file




## Notes:

- EEPROM Programming
  - Power off EL9800
  - Set PDI Selector to 0 ("OFF")
  - Power on EL9800
  - Copy the ESI file (My Slave.xml) to the TwinCAT ESI folder "c:\TwinCAT\3.1\Config\io\EtherCAT\"
  - Rebuild ESI cache (TWINCAT->EtherCAT Devices -> Reload Device Devicedescriptions)
  - Program "MySlave" to EEPROM
  - Power off EL9800
  - Set PDI Selector to 7 ("PIC (SPI)")
  - Connect the PIC Programmer with the Computer
  - Switch on the PIC Programmer (SW600)
  - Power on EL9800




## Notes:

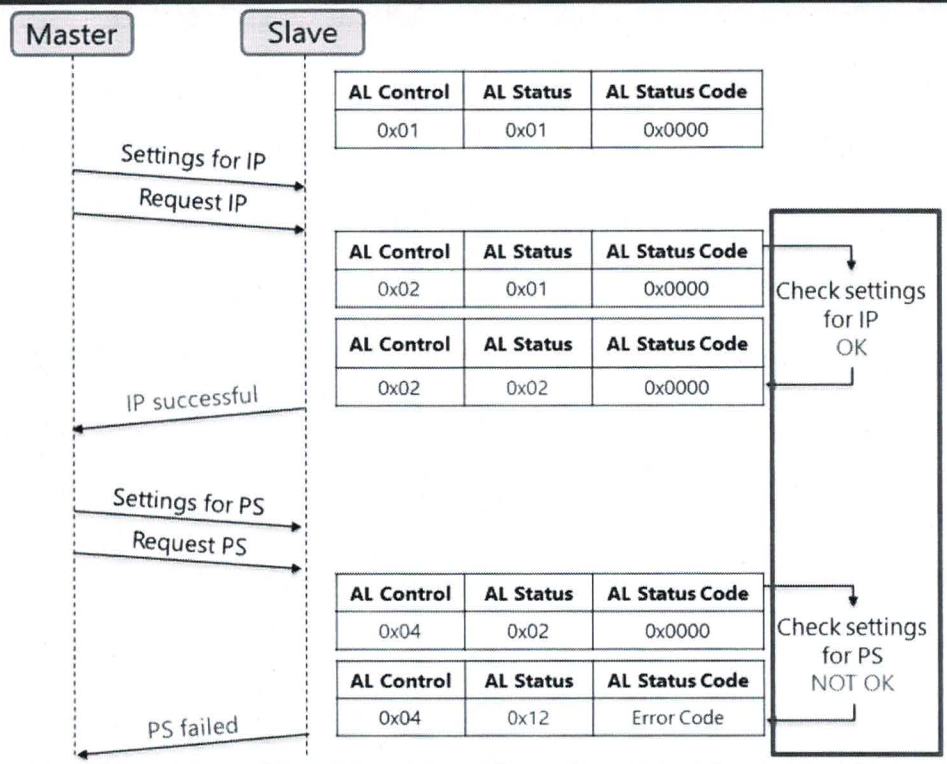
- Create MPLAB X IDE Project (and download binary)
- Detailed “Create Project” is described in EL9800 Application Note

- EtherCAT State machine
- Mailbox communication
- Process data communication
- Synchronization

Notes:

# ESM | Transition Procedure

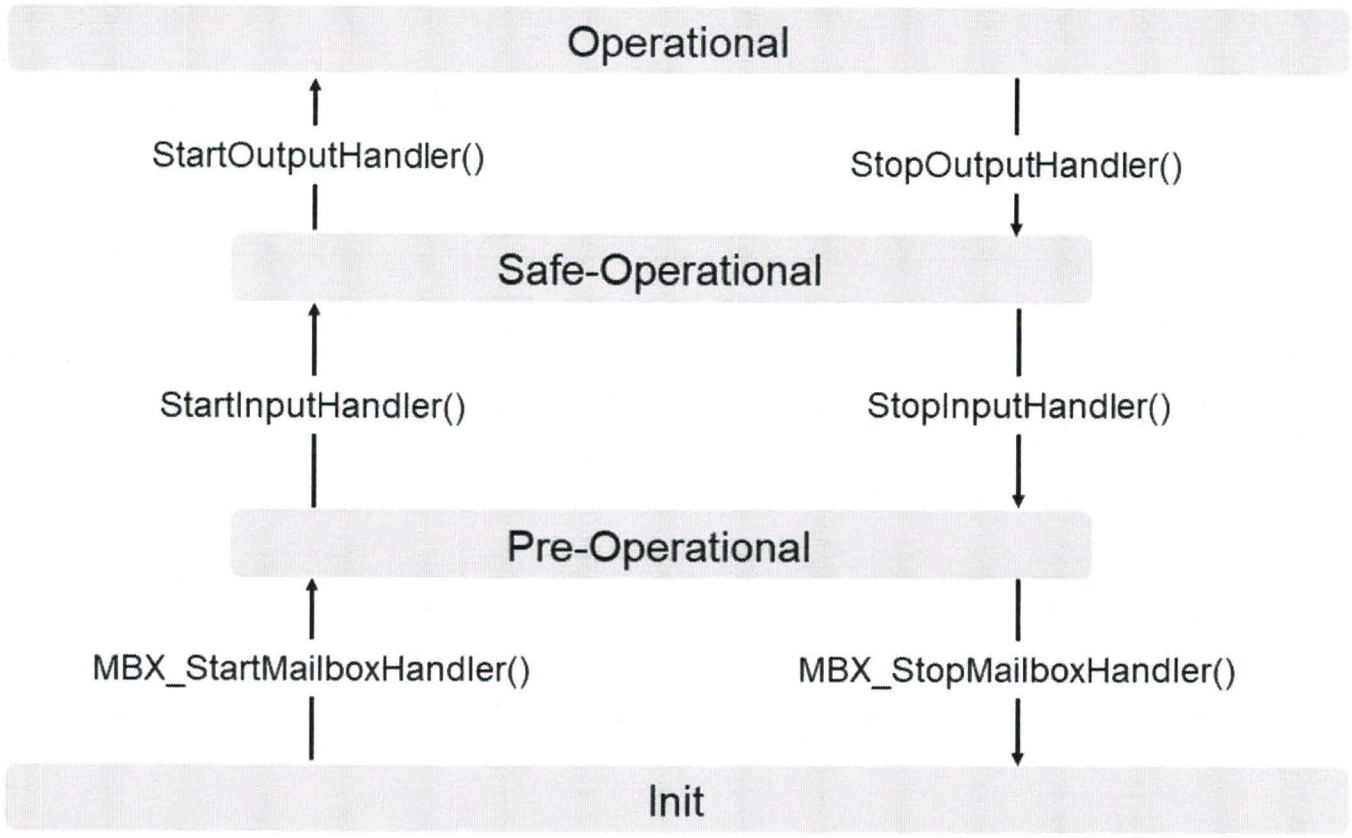
BECKHOFF




Notes:

# ESM (EtherCAT State Machine) | Generic

BECKHOFF

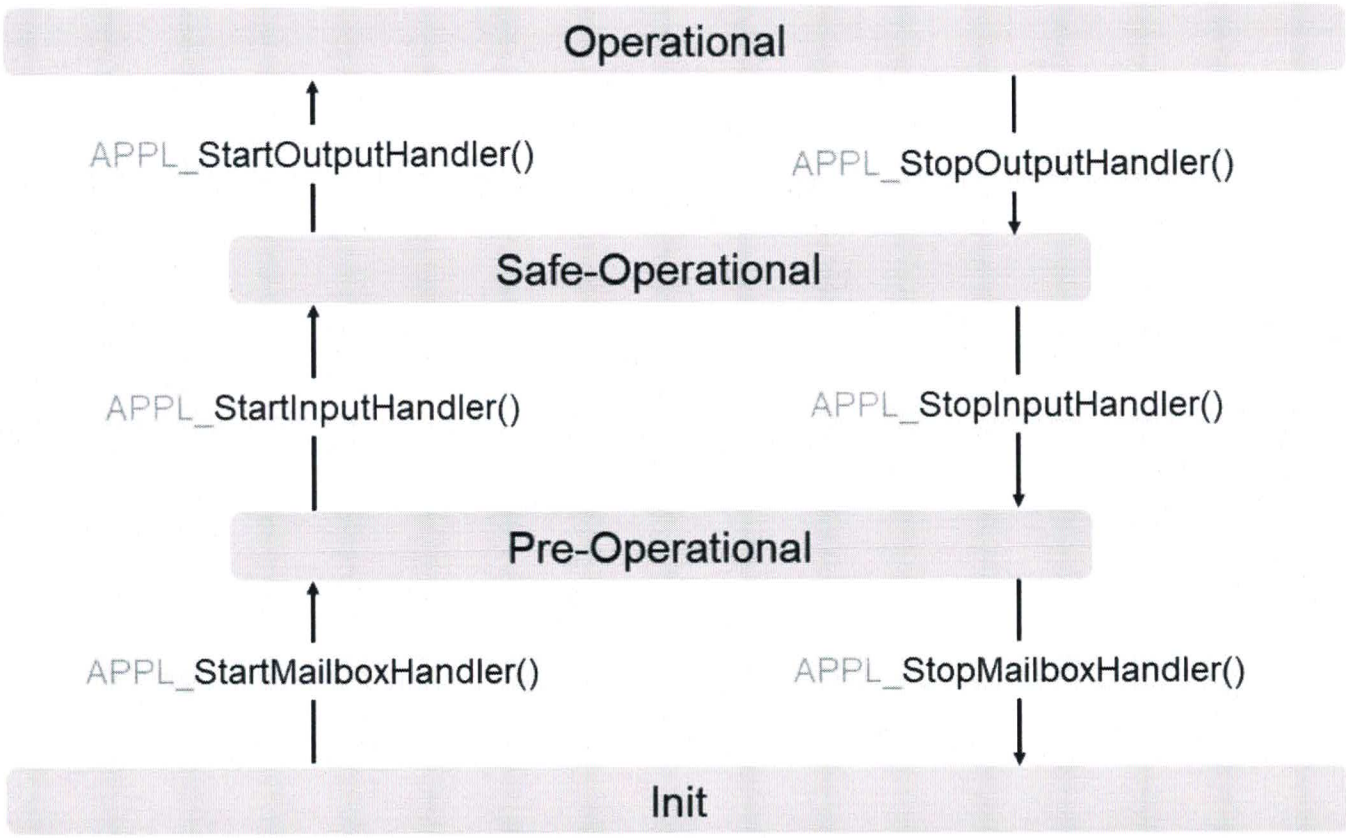



## Notes:

- Backward transitions over states are possible, the corresponding functions are called implicitly

# ESM (EtherCAT State machine) | Application

BECKHOFF




## Notes:

- Backward transitions over states are possible, the corresponding functions are called implicitly

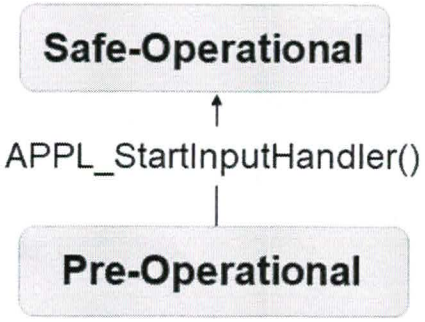
# ESM | Modify PreOP to SafeOP Transition

BECKHOFF

Switch 1

FALSE : PreOP to SafeOP accepted

TRUE : PreOP to SafeOP rejected with AL status Code 0x52




```
116
117  UINT16 APPL_StartInputHandler(UINT16 *pIntMask)
118  {
119      if(SWITCH_1)
120      {
121          return ALSTATUSCODE_EXT_HARDWARE_NOT_READY; /*0x52*/
122      }
123      else
124      {
125          return ALSTATUSCODE_NOERROR;
126      }
127  }
128
```

## Notes:

- File: MySlave.c
- Function : APPL_StartInputHandler
- MPLAB X supports autocomplete by pressing “Ctrl” + “Space”

Switch 2

FALSE : Normal Operation

TRUE : If the slave is in OP switch to SafeOP with AL Status Code 0x02

else the slave state remains unchanged

```
306 /*////////////////////////////////////////////////////////////////////
307 void APPL_Application(void)
308 {
309     if(SWITCH_2)
310     {
311         if(bLocalErrorFlag == FALSE)
312         {
313             ECAT_StateChange(STATE_SAFEOP,ALSTATUSCODE_NOMEMORY);
314         }
315     }
316     else
317     {
318         ECAT_StateChange(0,0);
319     }
320
321 #if _WIN32
```

## Notes:

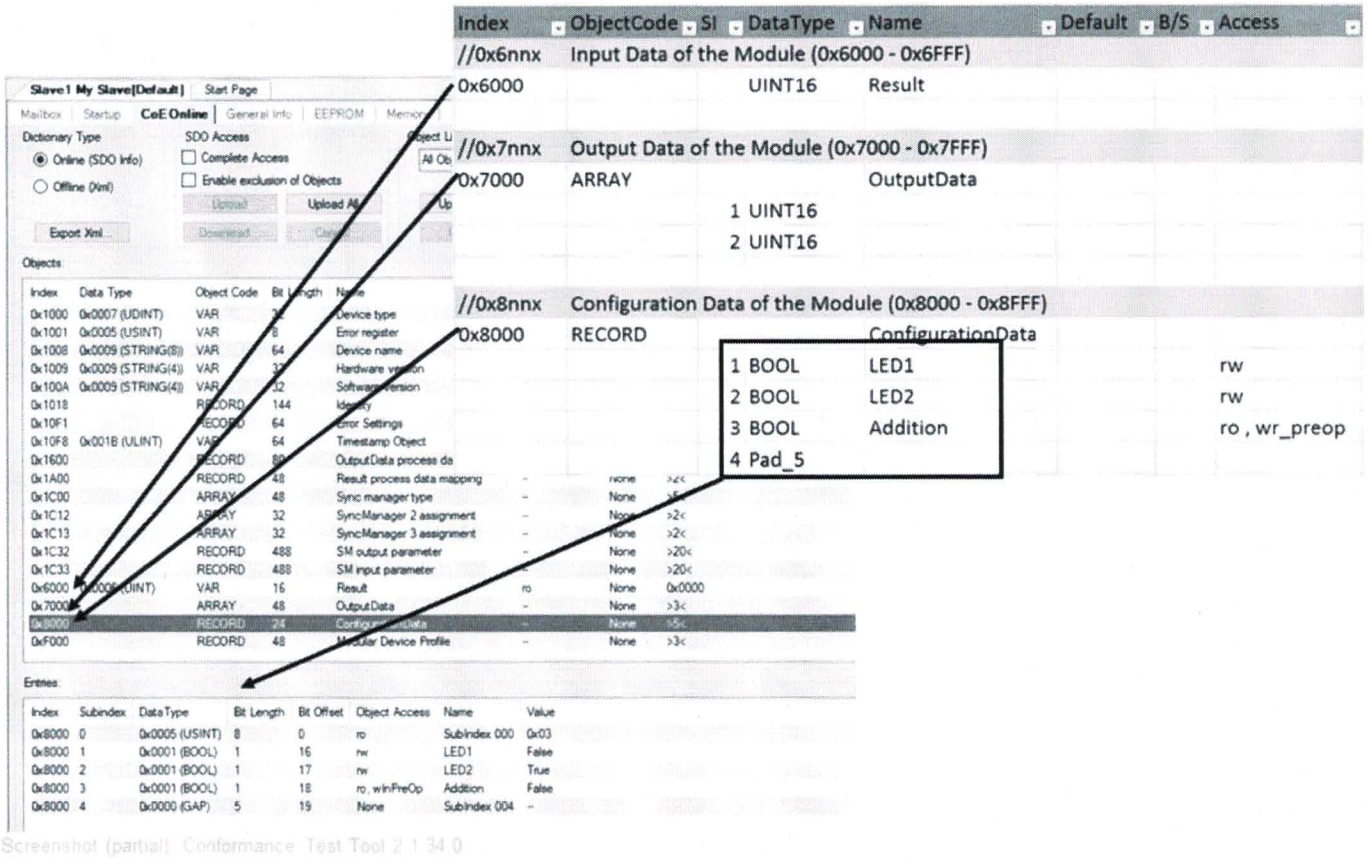
- File: MySlave.c
- Function : APPL_Application
- Only transitions to lower states are allowed (will be handled by ECAT_StateChange)
- ECAT_StateChange
  - 1. Parameter specifies the highest available state. In this example the slave still can be set to INIT,PreOP and SafeOP, a state request to OP would be rejected with “ALSTATUSCODE_NOMEMORY” (0x2).
  - 2. Parameter specifies the AL Status Code which will be returned
- If the local error is gone the ECAT_StateChange shall be called with both parameters set to 0

## Slave Application | Components

BECKHOFF

- ✓ EtherCAT State machine
- Mailbox communication
- Process data communication
- Synchronization

Notes:


Index ObjectCode SI DataType Name Default B/S Access
//0x6nnx Input Data of the Module (0x6000 - 0x6FFF)
0x6000 UINT16 Result
//0x7nnx Output Data of the Module (0x7000 - 0x7FFF)
0x7000 ARRAY OutputData
1 UINT16
2 UINT16
//0x8nnx Configuration Data of the Module (0x8000 - 0x8FFF)
0x8000 RECORD ConfigurationData
1 BOOL LED1 rw
2 BOOL LED2 rw
3 BOOL Addition ro , wr_preop
4 Pad_5
Slave1 My Slave[Default] Start Page
Mailbox Startup CoE Online General Info EEPROM
Dictionary Type
Online (SDO Info)
Offline (Xml)
SDO Access
Complete Access
Enable exclusion of Objects
Upload
Upload All
Export Xml...
Download
Objects
Entries
Screenshot (partial) Conformance Test Tool 2.1.34.0

## Notes:

| Index | ObjectCode | SI | DataType | Name | Default | B/S | Access |
|---|---|---|---|---|---|---|---|
| //0x6nnx | Input Data of the Module (0x6000 - 0x6FFF) | | | | | | |
| 0x6000 | | | UINT16 | Result | | | |
| //0x7nnx | Output Data of the Module (0x7000 - 0x7FFF) | | | | | | |
| 0x7000 | ARRAY | | | OutputData | | | |
| | | 1 | UINT16 | | | | |
| | | 2 | UINT16 | | | | |
| //0x8nnx | Configuration Data of the Module (0x8000 - 0x8FFF) | | | | | | |
| 0x8000 | RECORD | | | ConfigurationData | | | |
| | | 1 | BOOL | LED1 | | | rw |
| | | 2 | BOOL | LED2 | | | rw |
| | | 3 | BOOL | Addition | | | ro , wr_preop |
| | | 4 | Pad_5 | | | | |

```
TOBJECT    OBJMEM ApplicationObjDic[] = {
/* Object 0x1600 */
{NULL , NULL , 0x1600 , (DEFTYPE_PDOMAPPING , 2 | (OBJCODE_REC << 8))  , asEntryDesc0x1600 , aName0x1600 , &OutputDataProcessDataMapping0x1600 , NULL , NULL , 0x0000 },
/* Object 0x1A00 */
{NULL , NULL , 0x1A00 , (DEFTYPE_PDOMAPPING , 1 | (OBJCODE_REC << 8))  , asEntryDesc0x1A00 , aName0x1A00 , &ResultProcessDataMapping0x1A00      , NULL , NULL , 0x0000 },
/* Object 0x1C12 */
{NULL , NULL , 0x1C12 , (DEFTYPE_UNSIGNED16 , 1 | (OBJCODE_ARR << 8))  , asEntryDesc0x1C12 , aName0x1C12 , &sRxPDOassign                      , NULL , NULL , 0x0000 },
/* Object 0x1C13 */
{NULL , NULL , 0x1C13 , (DEFTYPE_UNSIGNED16 , 1 | (OBJCODE_ARR << 8))  , asEntryDesc0x1C13 , aName0x1C13 , &sTxPDOassign                      , NULL , NULL , 0x0000 },
/* Object 0x6000 */
{NULL , NULL , 0x6000 , (DEFTYPE_UNSIGNED16 , 0 | (OBJCODE_VAR << 8))  , &sEntryDesc0x6000 , aName0x6000 , &Result0x6000                      , NULL , NULL , 0x0000 },
/* Object 0x7000 */
{NULL , NULL , 0x7000 , (DEFTYPE_UNSIGNED16 , 2 | (OBJCODE_ARR << 8))  , asEntryDesc0x7000 , aName0x7000 , &OutputData0x7000                  , NULL , NULL , 0x0000 },
/* Object 0x8000 */
{NULL , NULL , 0x8000 , (DEFTYPE_RECORD , 4 | (OBJCODE_REC << 8))      , asEntryDesc0x8000 , aName0x8000 , &ConfigurationData0x8000        , NULL , NULL , 0x0000 },
/* Object 0xF000 */
{NULL , NULL , 0xF000 , (DEFTYPE_RECORD , 2 | (OBJCODE_REC << 8))      , asEntryDesc0xF000 , aName0xF000 , &ModularDeviceProfile0xF000     , NULL , NULL , 0x0010 },
{NULL , NULL , 0xFFFF , ( 0 , 0 ) , NULL, NULL, NULL, NULL}};
```

## Notes:

- The Object Dictionary is a list of slave application data which can be accessed by the master via CoE
- The object related variables and defines are located in [ApplicationName]Objects.h

a. Object Index
b. Object Description
c. Entry Description
d. Names
e. Memory / Variable

```
a                b                                                c                d                e
TOBJECT    OBJMEM ApplicationObjDic[] = {
/* Object 0x1600 */
(NULL , NULL ,  0x1600 , {DEFTYPE_PDOMAPPING , 2 | (OBJCODE_REC << 8)}  , asEntryDesc0x1600 , aName0x1600 , &OutputDataProcessDataMapping0x1600 , NULL , NULL , 0x0000 ),
/* Object 0x1A00 */
(NULL , NULL ,  0x1A00 , {DEFTYPE_PDOMAPPING , 1 | (OBJCODE_REC << 8)}  , asEntryDesc0x1A00 , aName0x1A00 , &ResultProcessDataMapping0x1A00 , NULL , NULL , 0x0000 ),
/* Object 0x1C12 */
(NULL , NULL ,  0x1C12 , {DEFTYPE_UNSIGNED16 , 3 | (OBJCODE_ARR << 8)}  , asEntryDesc0x1C12 , aName0x1C12 , &sRxPDOassign , NULL , NULL , 0x0000 ),
/* Object 0x1C13 */
(NULL , NULL ,  0x1C13 , {DEFTYPE_UNSIGNED16 , 1 | (OBJCODE_ARR << 8)}  , asEntryDesc0x1C13 , aName0x1C13 , &sTxPDOassign , NULL , NULL , 0x0000 ),
/* Object 0x6000 */
(NULL , NULL ,  0x6000 , {DEFTYPE_UNSIGNED16 , 0 | (OBJCODE_VAR << 8)}  , asEntryDesc0x6000 , aName0x6000 , &Result0x6000 , NULL , NULL , 0x0000 ),
/* Object 0x7000 */
(NULL , NULL ,  0x7000 , {DEFTYPE_UNSIGNED16 , 2 | (OBJCODE_ARR << 8)}  , asEntryDesc0x7000 , aName0x7000 , &OutputData0x7000 , NULL , NULL , 0x0000 ),
/* Object 0x8000 */
(NULL , NULL ,  0x8000 , {DEFTYPE_RECORD , 4 | (OBJCODE_REC << 8)}  , asEntryDesc0x8000 , aName0x8000 , &ConfigurationData0x8000 , NULL , NULL , 0x0000 ),
/* Object 0xF000 */
(NULL , NULL ,  0xF000 , {DEFTYPE_RECORD , 2 | (OBJCODE_REC << 8)}  , asEntryDesc0xF000 , aName0xF000 , &ModularDeviceProfile0xF000 , NULL , NULL , 0x0000 ),
(NULL , NULL ,  0xFFFF , {0 , 0} , NULL, NULL, NULL, NULL)};
```

## Notes:

- Detailed description of the Object Dictionary and the dictionary entries is available in the SSC Application Note : www.beckhoff.com/download/ethercat-development-products/
- The object related variables and defines are located in [ApplicationName]Objects.h

```
EtherCAT datagram(s): 'FPWR': Len: 128, Adp 0x3e9, Ado 0x1000, Wc 0
v EtherCAT datagram: Cmd: 'FPWR' (5), Len: 128, Adp 0x3e9, Ado 0x1000, Cnt 0
  v Header
      Cmd        : 5 (Configured address Physical Write)
      Index: 0xb4
      Slave Addr: 0x03e9
      Offset Addr: 0x1000
    > Length     : 128 (0x80) - No Roundtrip - Last Sub Command
      Interrupt: 0x0000
  v EtherCAT Mailbox Protocol:CoE SDO Req : 'Initiate Download' (1) Idx=0x8000 Sub=1
    v Header
        Length: 10
        Address: 0x0000
        .... ..00 = Priority: 0
        Type: CoE (CANopen over EtherCAT) (3)
        Counter: 6
    v CoE
        Number: 0
        Type: SDO Req (2)
      > SDO Req : 'Initiate Download' (1) Idx=0x8000 Sub=1
        Index: 0x8000
        SubIndex: 0x01
        Data: 0x01
    Data: 0000000000000000000000000000000000000000000000000000...
    Working Cnt: 0
```

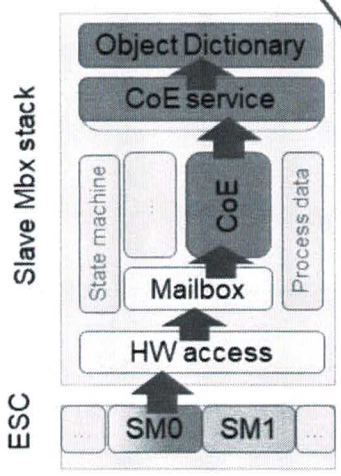
SyncManger 0 Buffer

| 0x1000 | Mailbox header | | | | | CoE protocol | | | | | | 0x1080 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10 | 0x0000 | 0 | 3 | 6 | 0 | 2 | 0x01 | 0x8000 | 0x1 | 0x1 | |
| | Length | Address | Priority | Type | Counter | Number | Type | OpCode | ObjIndex | Subindex | Data | |

## Notes:

- Master send request
- Write to Out mailbox (SM0)
- SDO Info of object 0x1008
- Write last byte of SM buffer to set write complete bit (0x220.8)

```
418 #ifdef _OBJD_
419 TOBJECT    OBJMEM ApplicationObjDic[] = {
420 /* Object 0x1600 */
421 {NULL , NULL ,  0x1600 , {DEFTYPE_PDOMAPPING , 2 | (OBJCODE_REC << 8)} , asEntryDesc0x1600 , aName0x1600 , &OutputDataProcessDataMapping(
422 /* Object 0x1A00 */
423 {NULL , NULL ,  0x1A00 , {DEFTYPE_PDOMAPPING , 1 | (OBJCODE_REC << 8)} , asEntryDesc0x1A00 , aName0x1A00 , &ResultProcessDataMapping0x1A(
424 /* Object 0x1C12 */
425 {NULL , NULL ,  0x1C12 , {DEFTYPE_UNSIGNED16 , 1 | (OBJCODE_ARR << 8)} , asEntryDesc0x1C12 , aName0x1C12 , &sRxPDOassign , NULL , NULL ,
426 /* Object 0x1C13 */
427 {NULL , NULL ,  0x1C13 , {DEFTYPE_UNSIGNED16 , 1 | (OBJCODE_ARR << 8)} , asEntryDesc0x1C13 , aName0x1C13 , &sTxPDOassign , NULL , NULL ,
428 /* Object 0x6000 */
429 {NULL , NULL ,  0x6000 , {DEFTYPE_UNSIGNED16 , 0 | (OBJCODE_VAR << 8)} , &sEntryDesc0x6000 , aName0x6000 , &Result0x6000 , NULL , NULL ,
430 /* Object 0x7000 */
431 {NULL , NULL ,  0x7000 , {DEFTYPE_UNSIGNED16 , 3 | (OBJCODE_ARR << 8)} , asEntryDesc0x7000 , aName0x7000 , &OutputData0x7000 , NULL , NU
432 /* Object 0x8000 */
433 {NULL , NULL ,  0x8000 , {DEFTYPE_RECORD , 4 | (OBJCODE_REC << 8)} , asEntryDesc0x8000 , aName0x8000 , &ConfigurationData0x8000 , NULL ,
434 /* Object 0xF000 */
435 {NULL , NULL ,  0xF000 , {DEFTYPE_RECORD , 2 | (OBJCODE_REC << 8)} , asEntryDesc0xF000 , aName0xF000 , &ModularDeviceProfile0xF000 , NULL
436 {NULL,NULL, 0xFFFF, {0,0}, NULL, NULL, NULL, NULL}};
437 #endif    //#ifdef _OBJD_
438 #undef PROTO
```




| | |
|---|---|
| 0x1 | Data |
| 0x1 | Subindex |
| 0x8000 | Obj Index |
| 0x01 | |
| 2 | |
| 0 | |
| 6 | |
| 3 | |
| 0 | |
| 0x0000 | |
| 10 | |

```
335 #ifndef _MY_SLAVE_OBJECTS_H_
336 /**
337  * \brief Object structure
338  */
339 typedef struct OBJ_STRUCT_PACKED_START {
340  UINT16 u16SubIndex0;
341  BOOLEAN(LED1); /* Subindex1 - LED1 */
342  BOOLEAN(LED2); /* Subindex2 - LED2 */
343  BOOLEAN(Addition); /* Subindex3 - Addition */
344  ALIGN5(SI4) /* Subindex4 */
345 } OBJ_STRUCT_PACKED_END
346 TOBJ8000;
347 #endif //#ifndef _MY_SLAVE_OBJECTS_H_
348
349 /**
350  * \brief Object variable
351  */
352 PROTO TOBJ8000 ConfigurationData0x8000
353 #if defined(_MY_SLAVE_) && (_MY_SLAVE_ == 1)
354 =(0,0,0,0,0)
355 #endif
356 ;
357 /** @}*/
```

## Notes:

```
307 void APPL_Application(void)
308 {
309     if(SWITCH_2)
310     {
311         if(bLocalErrorFlag == FALSE)
312         {
313             ECAT_StateChange(STATE_SAFEOP,ALSTATUSCODE_NOMEMORY);
314         }
315     }
316     else
317     {
318         ECAT_StateChange(0,0);
319     }
320
321     LED_1 = ConfigurationData0x8000.LED1;
322     LED_2 = ConfigurationData0x8000.LED2;
323
```

## Notes:

- The object variables are named as follows : [DefinedName]0x[Index]
- Object values are used to set the hardware LED in the application

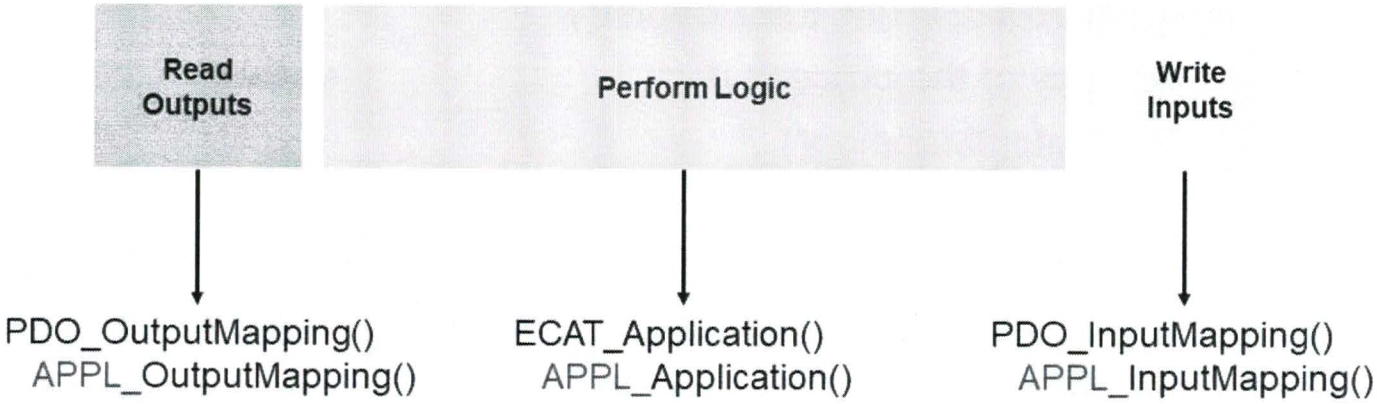
- ✓ EtherCAT State machine
- ✓ Mailbox communication
- Process data communication
- Synchronization

Notes:

# Process data handling

BECKHOFF

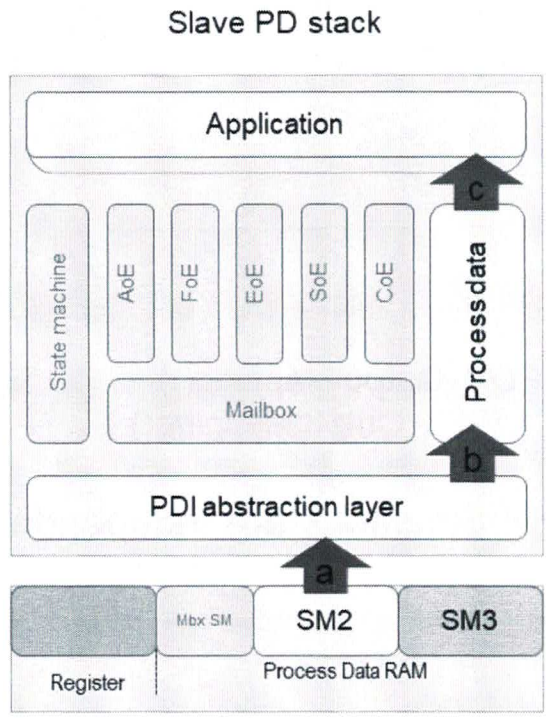
- 3 Methods are involved




- Functions are independent on the activated/supported Synchronization mode

Notes:

- PDO_OutputMapping()
  a. HW_EscRead()/HW_EscReadIsr()
  b. Local copy of the process data
  c. APPL_OutputMapping()




Notes:

| Index | ObjectCode | SI | DataType | Name |
|---|---|---|---|---|
| //0x6nnx | Input Data of the Module (0x6000 - 0x6FFF) | | | |
| 0x6000 | | | UINT16 | Result |
| //0x7nnx | Output Data of the Module (0x7000 - 0x7FFF) | | | |
| 0x7000 | ARRAY | | | OutputData |
| | | 1 | UINT16 | |
| | | 2 | UINT16 | |

```
typedef struct OBJ_STRUCT_PACKED_START {
UINT16    u16SubIndex0;   /**< \brief Subindex 0 */
UINT16 aEntries[2];       /**< \brief Subindex 1 - 2 */
} OBJ_STRUCT_PACKED_END
TOBJ7000;
#endif //#ifndef _MY_SLAVE_OBJECTS_H_

/**
 * \brief Object variable
 */
PROTO TOBJ7000 OutputData0x7000
```

```
286  void APPL_OutputMapping(UINT16* pData)
287  {
288      memcpy(OutputData0x7000.aEntries,pData,(SIZEOF(OutputData0x7000)-2));
289  }
```

Notes:

- Adapt Code

```
void APPL_Application(void)
{
    if(SWITCH_2)
    {
        if(bLocalErrorFlag == FALSE)
        {
            ECAT_StateChange(STATE_SAFEOP,ALSTATUSCODE_NOMEMORY);
        }
    }
    else
    {
        ECAT_StateChange(0,0);
    }

    if(ConfigurationData0x8000.Addition)
    {
        Result0x6000 = OutputData0x7000.aEntries[0] + OutputData0x7000.aEntries[1];
    }
    else
    {
        Result0x6000 = OutputData0x7000.aEntries[0] - OutputData0x7000.aEntries[1];
    }

    LED_1 = ConfigurationData0x8000.LED1;
    LED_2 = ConfigurationData0x8000.LED2;
```

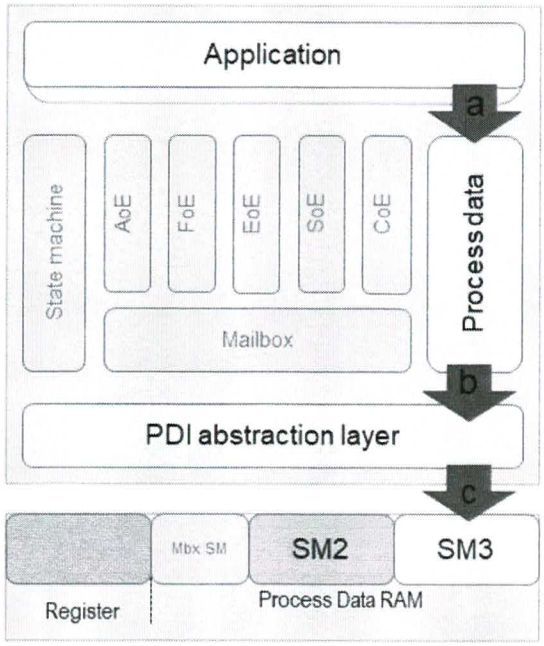
Notes:

# Process data handling | Inputs (Slave -> Master)

BECKHOFF

- PDO_InputMapping()
  a. APPL_InputMapping()
  b. Local copy of the process data
  c. HW_EscWrite()/HW_EscWriteIsr()

Slave PD stack




Notes:

| Index | ObjectCode | SI | DataType | Name |
|---|---|---|---|---|
| //0x6nnx | Input Data of the Module (0x6000 - 0x6FFF) | | | |
| 0x6000 | | | UINT16 | Result |
| //0x7nnx | Output Data of the Module (0x7000 - 0x7FFF) | | | |
| 0x7000 | ARRAY | | | OutputData |
| | | 1 | UINT16 | |
| | | 2 | UINT16 | |

```
243  /**
244   * \brief Object variable
245   */
246  PROTO UINT16 Result0x6000;
```

```
270  void APPL_InputMapping(UINT16* pData)
271  {
272      *pData = Result0x6000;
273  }
```

Notes:

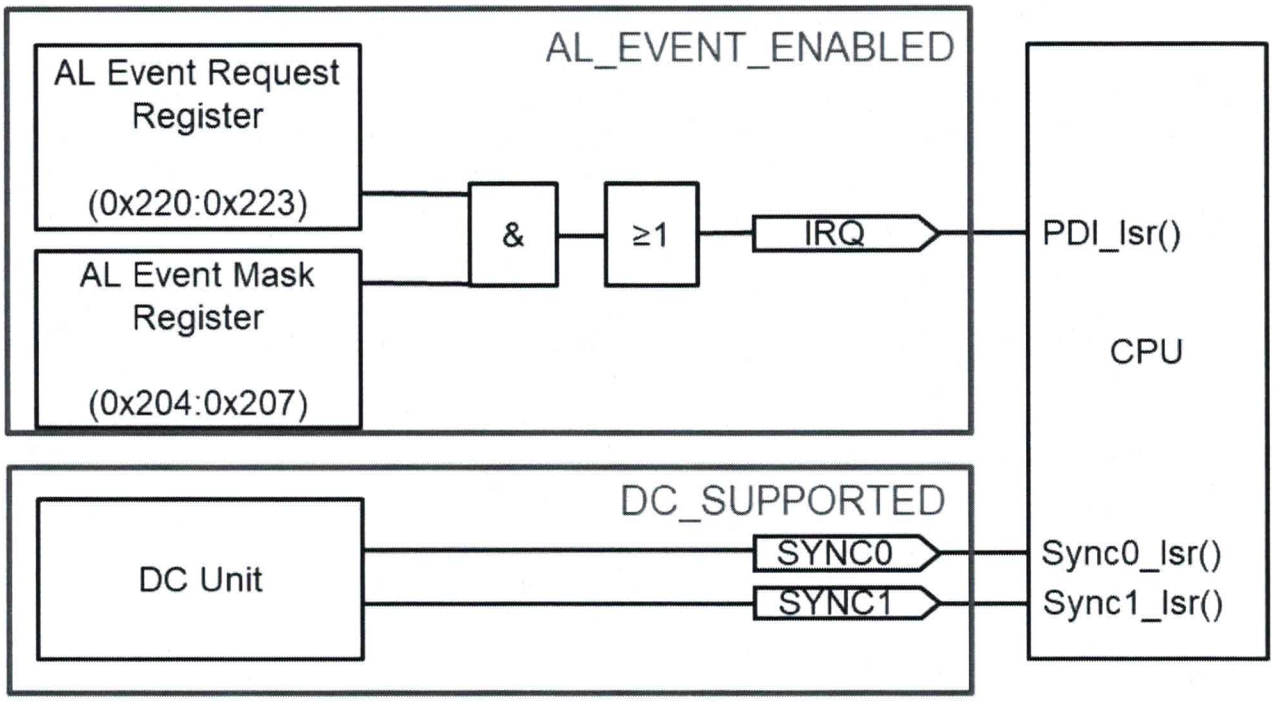
- Adapt Code and show behaviour

## Slave Application | Components

BECKHOFF

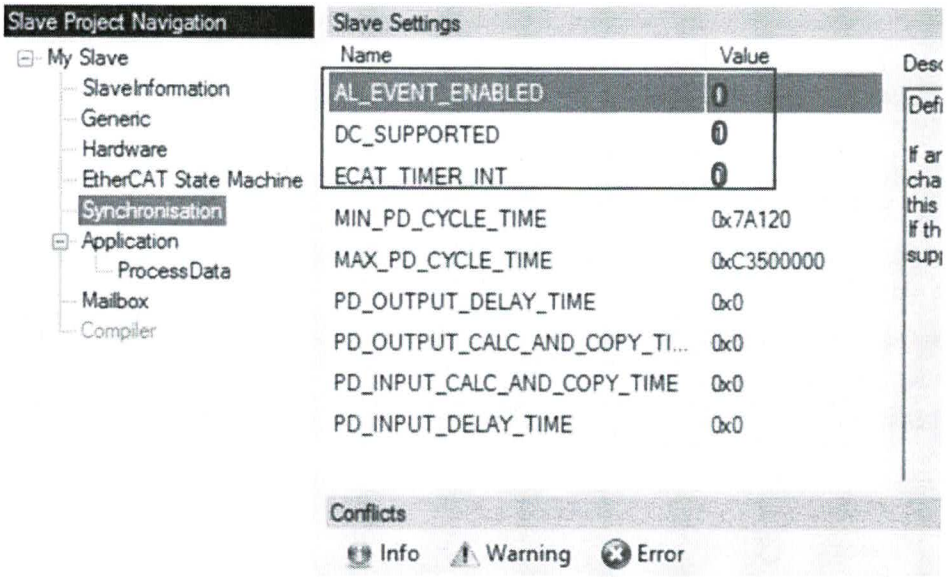
- ✓ EtherCAT State machine
- ✓ Mailbox communication
- ✓ Process data communication
- ▪ Synchronization

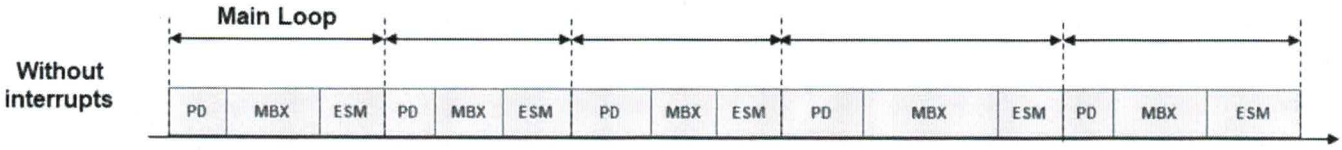
Notes:




## Notes:

- The defines “AL_EVENT_ENABLED” and “DC_SUPPORTED” are available in the Synchronization settings of the SSC Tool
- AL Event Mask :
  - by default the SSC only configures SM2 event (or SM3 for input only devices),
  - all other events mapped to the AL Event register are polled in the MainLoop (low priority)



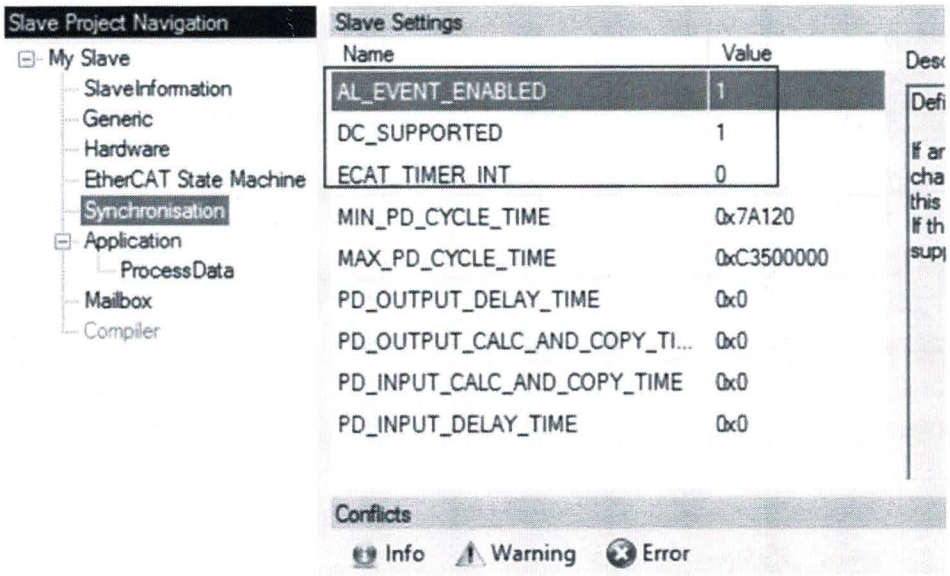



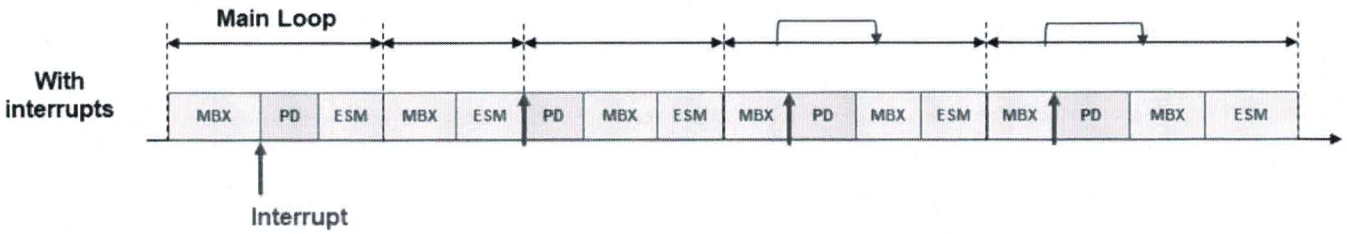

## Notes:

- Interrupts and Synchronization should be disabled for initial development steps
  - These functions can be enabled when the basic application is implemented

- ECAT_TIMER_INT: controls if the ticks of the CPU internal timer shall be handled via in ISR or by polling. The timer is required for watchdogs and Status LED control

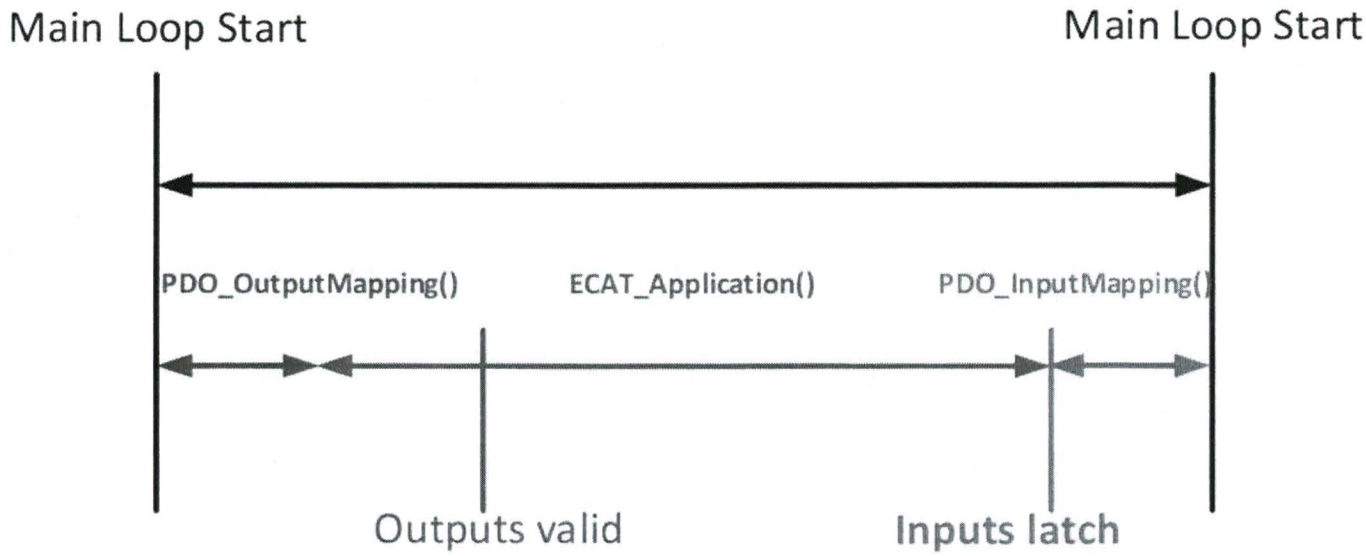
- PD : Process data handling
- MBX : Mailbox handling
- ESM : EtherCAT State Machine



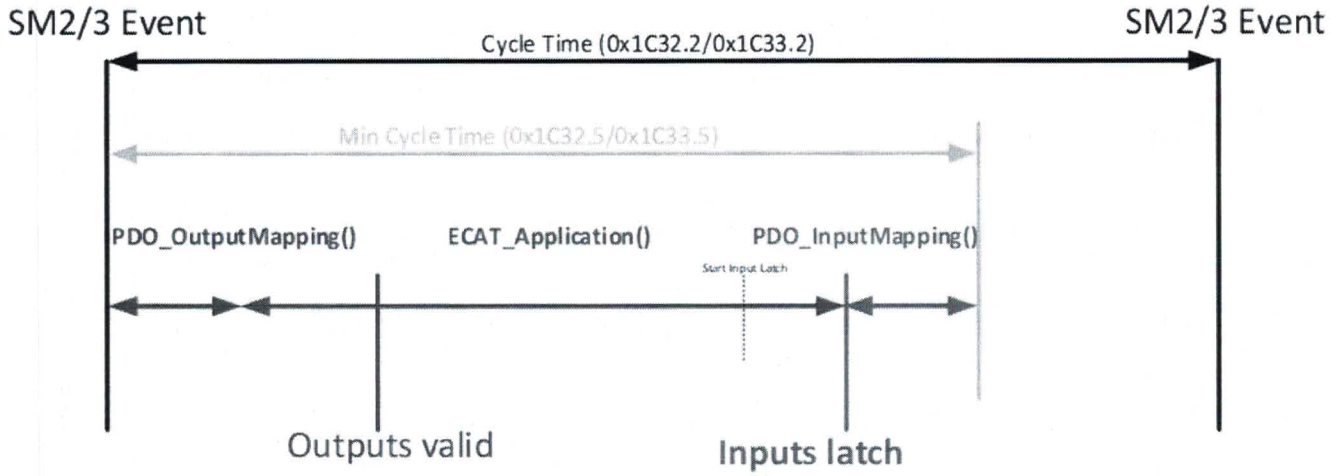

## Notes:

- PD : Process data handling
- MBX : Mailbox handling
- ESM : EtherCAT State Machine



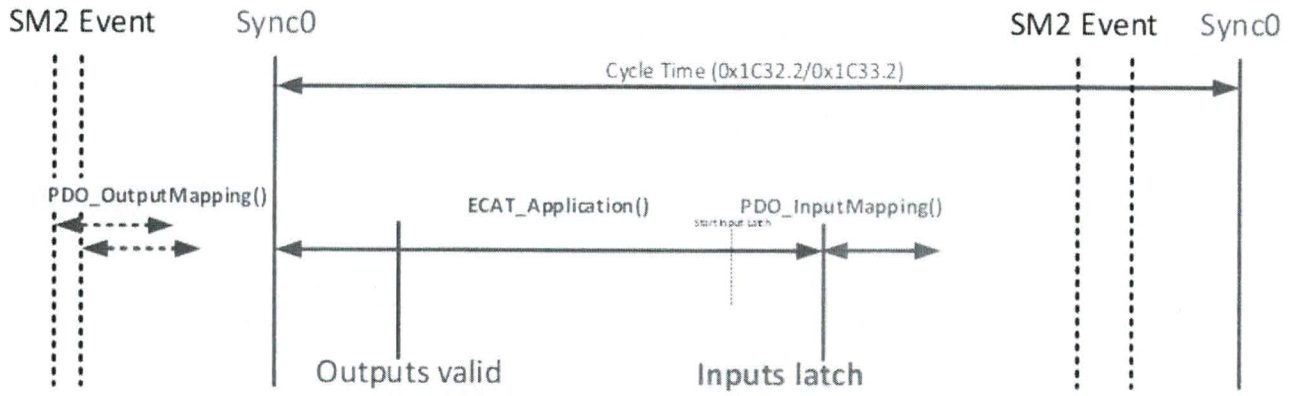

Notes:

- AL_EVENT_ENABLED = 0
- DC_SUPPORTED = 0




## Notes:

- AL_EVENT_ENABLED = 1
- DC_SUPPORTED = 0




## Notes:

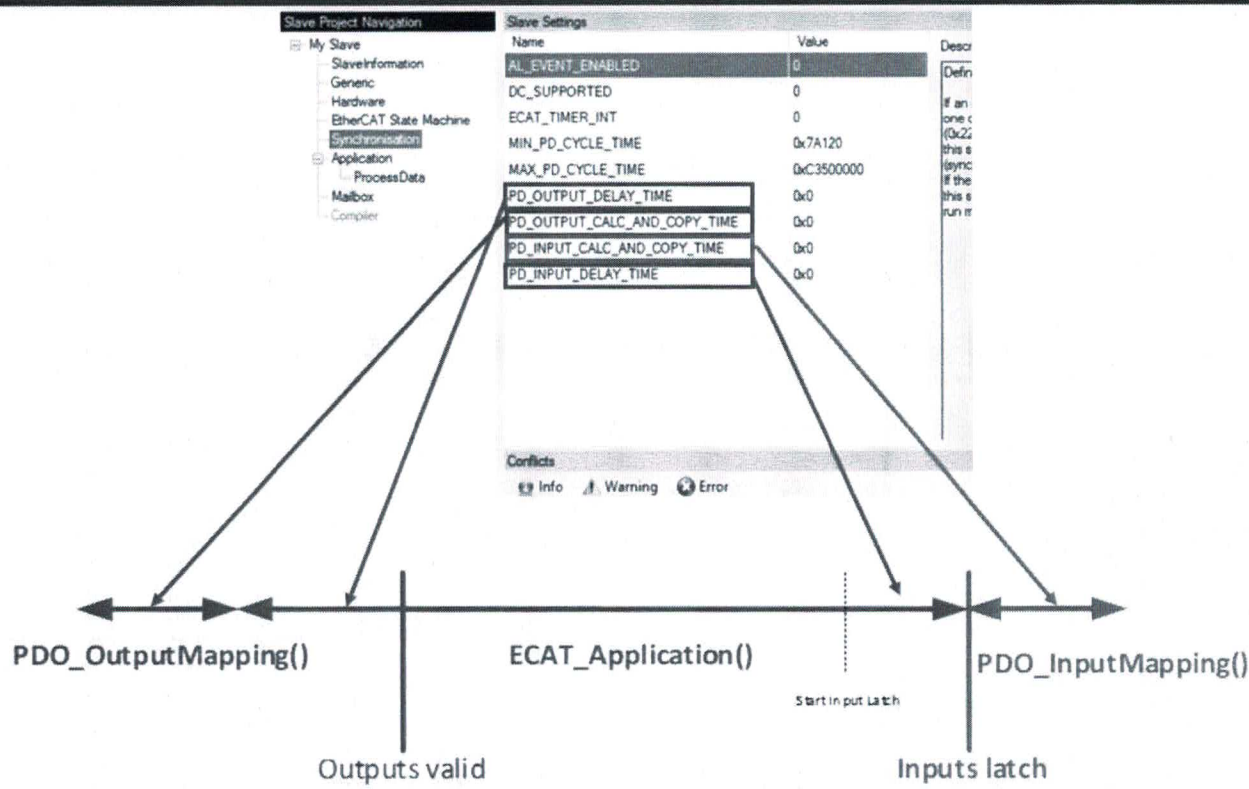
- AL_EVENT_ENABLED = 1
- DC_SUPPORTED = 1

- Sync Activation Register (0x981): Bit0, Bit1 = 1 (the register is set by the master, if it is not set the stack is running in SM Sync Mode, see previous slide).

- Further supported Sync Modes are listed in chapter "Synchronization" in the SSC Application Note : www.beckhoff.com/download/ethercat-development-products/

# Synchronization | Timing Parameter

BECKHOFF




Notes:

## Slave Application | Components

BECKHOFF

- ✓ EtherCAT State machine
- ✓ Mailbox communication
- ✓ Process data communication
- ✓ Synchronization

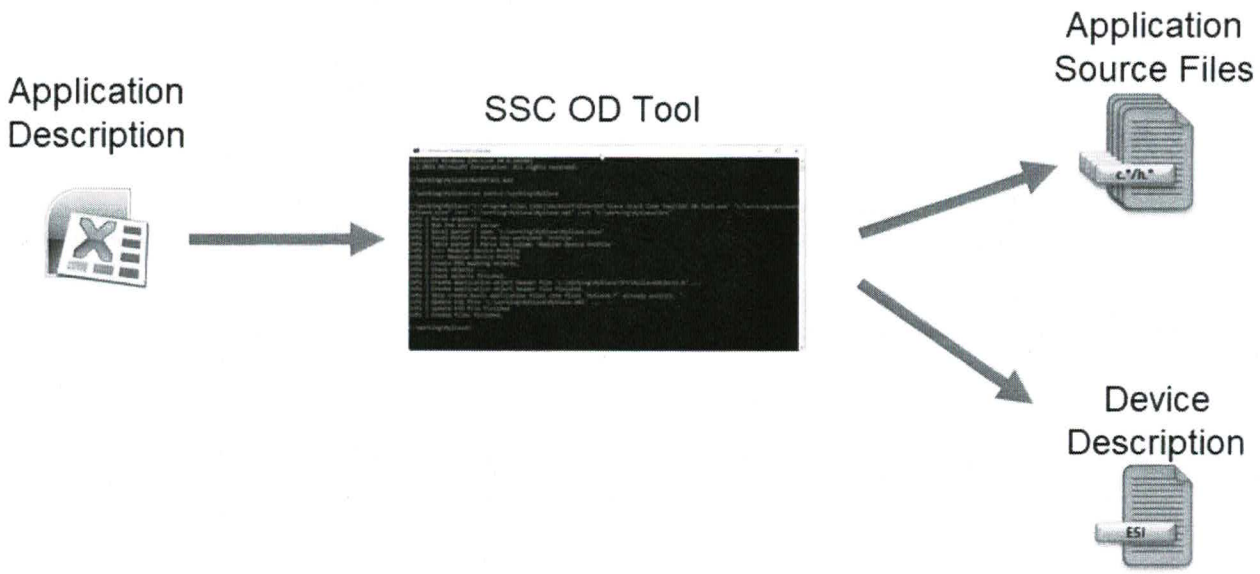
Notes:

| Index | ObjectCo | SI | DataType | Name | De | B/S | Access |
|---|---|---|---|---|---|---|---|
| //0x6nnx | Input Data of the Module (0x6000 - 0x6FFF) | | | | | | |
| 0x6000 | | | UINT16 | Result | | | |
| 0x6001 | | | UINT32 | Timestamp | | | |
| | | | | | | | |
| //0x7nnx | Output Data of the Module (0x7000 - 0x7FFF) | | | | | | |
| 0x7000 | ARRAY | | | OutputData | | | |
| | | 1 | UINT16 | | | | |
| | | 2 | UINT16 | | | | |
| //0x8nnx | Configuration Data of the Module (0x8000 - 0x8FFF) | | | | | | |
| 0x8000 | RECORD | | | ConfigurationData | | | |
| | | 1 | BOOL | LED1 | | | rw |
| | | 2 | BOOL | LED2 | | | rw |
| | | 3 | BOOL | Addition | | | ro, wr_preop |
| | | 4 | pad_5 | | | | |

Notes:

# Modify Application | Generate new files

BECKHOFF




## Notes:

- The SSC OD Tool is located in the SSC Tool installation folder(e.g. “c:\Program Files (x86)\Beckhoff\EtherCAT Slave Stack Code Tool\”)
- The command line parameter are described in the SSC Application Note (www.beckhoff.com/download/ethercat-development-products/)

```
void APPL_InputMapping(UINT16* pData)
{
    *pData = Result0x6000;
    pData++;
    memcpy(pData,&Timestamp0x6001,SIZEOF(Timestamp0x6001));
}
```

Notes:

```
void APPL_Application(void)
{
    if(ConfigurationData0x8000.Addition)
    {
        Result0x6000 = OutputData0x7000.aEntries[0] + OutputData0x7000.aEntries[1];
    }
    else
    {
        Result0x6000 = OutputData0x7000.aEntries[0] - OutputData0x7000.aEntries[1];
    }

    if(bDcRunning)
    {
        HW_EscReadDWordIsr(Timestamp0x6001,0x910);
    }
```

Notes:

**Beckhoff Automation GmbH & Co. KG**
Huelshorstweg 20
33415 Verl
Germany

Phone: +49 911 54056-26
E-Mail: EtherCATSSC@beckhoff.com
Web: www.beckhoff.com

**EtherCAT Technology Group**
Ostendstr. 196
90482 Nuremberg
Germany

Phone: +49 911 54056-20
E-Mail: info@ethercat.org
Web: www.ethercat.org

© Beckhoff Automation GmbH & Co. KG

All images are protected by copyright. The use and transfer to third parties is not permitted.

Beckhoff®, TwinCAT®, EtherCAT®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this presentation may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

The information provided in this presentation contains merely general descriptions or characteristics of performance which in case of actual application do not always apply as described or which may change as a result of further development of the products. An obligation to provide the respective characteristics shall only exist if expressively agreed in the terms of contract.

## Notes: