pbsSoftLogic EtherCAT Master Driver Configuration

May 2026

Ver. 1.0

1 – Introduction

This document provides a detailed description of the **pbsSoftLogic EtherCAT Master Driver**. The driver is fully compatible with EtherCAT I/O devices from any manufacturer, including Beckhoff, Omron, WAGO, and other ETG-compliant vendors.

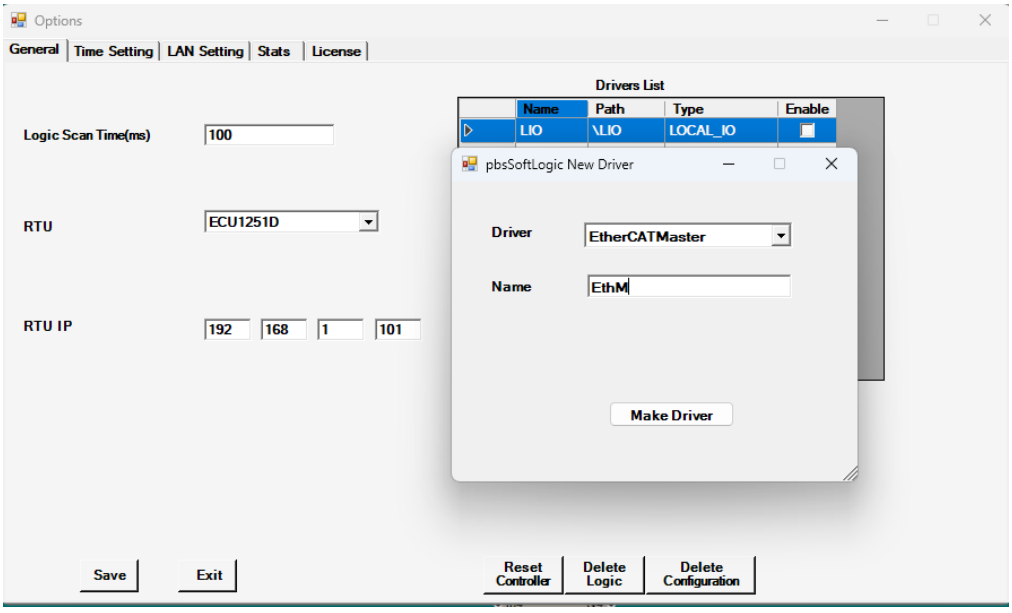
Using this driver, any Linux-based RTU—such as the ECU1251D, ECU150, BeagleBone Black, or Raspberry-Pi-based controllers—can interface directly with EtherCAT I/O modules without requiring special hardware.

The EtherCAT Master supports both **PDO cyclic Read/Write** operations and **CoE (CAN over EtherCAT)** access. All PDO and CoE objects are exposed to the pbsSoftLogic runtime through user-defined **pbsSoftLogic Tags**, which provide a unified method for reading, writing, and configuring EtherCAT data.

2 – Adding the EtherCAT Master Driver to a Project

To add the EtherCAT Master Driver, open Project Settings, right-click the Driver List, and select Add New Driver. Choose EtherCATMaster, assign a unique driver name, and click Make Driver.

The IDE automatically generates the default configuration and inserts the EtherCAT Master Driver into the project structure, making it ready for further setup and tag definition.



The pbsSoftLogic IDE creates a dedicated folder for each driver inside the project directory. Within this folder, two configuration files are generated: **Options.xml**, which contains all communication parameters, and **EtherCATTags.xml**, which stores every tag defined for the EtherCAT Master Driver.

You may edit these files using any external text editor, such as Notepad++, or by using the built-in editor in the pbsSoftLogic IDE. To open the editor, right-click the driver name and select **Edit**. The integrated editor displays both configuration files in a simple interface, organized into two separate tabs for convenient navigation. Figure 2 , 3

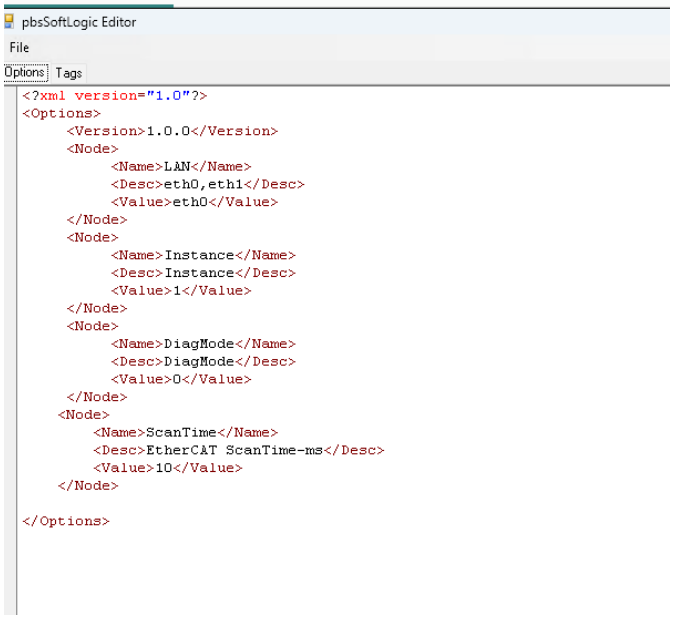


Figure -2

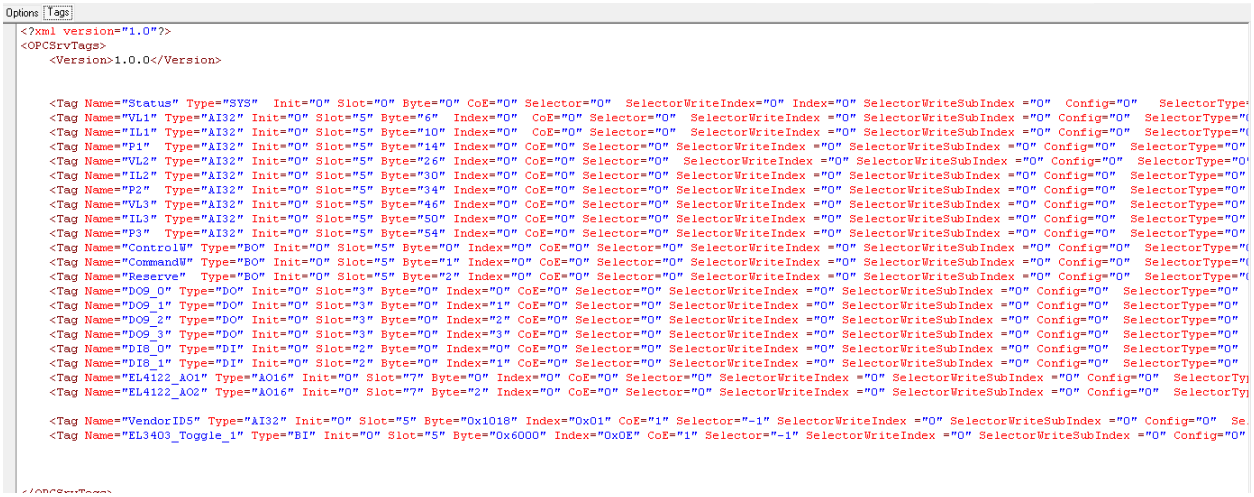


Figure -3

Communication parameters

LAN Specifies the network interface on the RTU that is connected to the EtherCAT devices. Enter the exact interface name (e.g., eth0, enp1s0, etc.).

DiagMode Enables or disables diagnostic output.

- When set to **1**, EtherCAT diagnostic information is displayed when the pbsSoftLogic kernel is executed manually on the RTU.
- When set to **0**, diagnostic output is disabled.

ScanTime Defines the cyclic read/write interval for PDO communication. If this value is set above **100 ms**, the watchdog timer of the EtherCAT coupler or adapter may trigger. Always verify the watchdog timeout value in the device's technical documentation to ensure compatibility.

Instance An internal parameter used by the runtime. No user configuration is required.

EtherCAT Tag Attributes

Each EtherCAT Tag in pbsSoftLogic includes the following attributes:

Name
Specifies the tag identifier. The final tag name used inside pbsSoftLogic is a combination of the **Driver Name** and the **Tag Name**, ensuring uniqueness across the project.

Type
Defines the data type and access direction. The following types are supported:

- **BI** — Reads a byte value from an input PDO or CoE object.
- **DI** — Reads a digital (bit) value from an input PDO or CoE object.
- **AI16** — Reads a 16-bit signed integer from an input PDO or CoE object.
- **AI32** — Reads a 32-bit signed integer from an input PDO or CoE object.
- **BO** — Writes a byte value to a PDO or CoE object.
- **DO** — Writes a digital (bit) output to a PDO or CoE object.
- **AO16** — Writes a 16-bit signed integer to a PDO or CoE object.
- **AO32** — Writes a 32-bit signed integer to a PDO or CoE object.

Init

Defines the initial value of the tag. This is typically used for EtherCAT configuration parameters or for initializing output tags at startup.

Slot

Each EtherCAT device in the network is automatically assigned a **Slot Number**. The master is always assigned **Slot 0**, while couplers, adapters, and I/O modules receive sequential slot numbers starting from **1**.

A sample EtherCAT network configuration using Beckhoff I/O modules is illustrated in *Figure 4*.

```
BEFORE OP: Slave 1 (EK1100) state=0x12, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 2 (EL1008) state=0x12, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 3 (EL2024) state=0x12, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 4 (EL2004) state=0x12, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 5 (EL3403-0010) state=0x02, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 6 (EL3403-0010) state=0x02, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 7 (EL4122) state=0x02, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 8 (EL4122) state=0x02, ALstatuscode=0x0000, Ibytes=0, Obytes=0
BEFORE OP: Slave 9 (EL3632) state=0x02, ALstatuscode=0x0000, Ibytes=0, Obytes=0
```

Figure -4

PDO and CoE Concepts, Differences, and Applications

EtherCAT devices expose their data through two primary communication mechanisms: **PDO (Process Data Objects)** and **CoE (CAN over EtherCAT)**. PDOs are used for **real-time cyclic data exchange**, providing fast, deterministic transfer of input and output values during each EtherCAT scan cycle. Typical PDO data includes digital inputs, analog measurements, counters, and actuator outputs—values that must be updated continuously with minimal latency. In contrast, CoE provides **asynchronous, mailbox-based access** to the device's Object Dictionary. CoE is used for configuration parameters, calibration values, diagnostic information, and any data that does not require cyclic real-time updates. While PDOs are optimized for speed and determinism, CoE is optimized for flexibility and completeness. In practical applications, PDOs handle the operational I/O data, whereas CoE is used for device setup, reading extended measurements, adjusting parameters, and performing maintenance tasks. Together, PDO and CoE provide a complete communication model for both real-time control and device management within the EtherCAT network.

SDO, or **Service Data Object**, is the mailbox-based communication mechanism used in EtherCAT to access a device's **Object Dictionary**. Unlike PDOs, which operate cyclically and in real time, SDO communication is **asynchronous** and designed for reading and writing configuration parameters, calibration values, diagnostic information, and any non-cyclic data. Each SDO transaction targets a specific **Index** and **Subindex** within the Object Dictionary, allowing precise access to device settings and extended measurement values. SDO communication is essential

during device initialization, parameter tuning, and maintenance operations, where deterministic timing is not required but full access to device internals is necessary. In pbsSoftLogic, SDO operations are performed automatically by the EtherCAT Master Driver when a tag is defined as a **CoE object**, ensuring clean separation between real-time logic and configuration tasks.

The **CoE** attribute determines whether a tag accesses a **PDO** entry or a **CoE (SDO)** object.

- **CoE = "1"** — The tag is mapped to a **CoE Object** and is read or written using mailbox-based SDO communication.
- **CoE = "0"** — The tag is mapped to a **PDO Object** and is exchanged cyclically during the EtherCAT scan.

This attribute allows each tag to explicitly define the communication method used by the EtherCAT Master Driver.

Here is your paragraph rewritten in a **clean, precise, professional, manual-grade style**, fully aligned with the rest of your EtherCAT Master documentation:

Byte
Specifies the starting byte offset within the PDO buffer or the CoE Index. If the value begins with 0x, it is interpreted as a hexadecimal number.

Index
For **PDO tags**, this field is used only with **DI** (digital input) types and represents the **bit number** within the specified byte.
For **CoE tags**, the Index field corresponds to the **SubIndex** of the CoE Object Dictionary entry.

Examples

- **Reading an 8-channel digital input module from PDO**
  If the digital input data is located at **byte 10**, then:
  - Channel 0 → Byte = 10, Index = 0
  - Channel 7 → Byte = 10, Index = 7
- **Reading a CoE object**
  To read object **0x6000:0x1E**, set:
  - Byte = 0x6000

  - Index = 0x1E

- **Reading a 16-bit analog input from PDO**
  If the value starts at **byte 20**, set:
  - Type = AI16
  - Byte = 20
  - Index is **not used** for AI16/AI32 types.

Selector, SelectorWriteIndex, SelectorWriteSubIndex, SelectorType

Some EtherCAT devices require a **selector write operation** before a CoE object can be read. In these cases, the device exposes multiple measurement values through a single CoE register, and the selector determines which value is returned.

The Beckhoff **EL3403** three-phase power measurement terminal is a typical example. To read values such as apparent power, reactive power, energy, power factor, or frequency, the EtherCAT Master must first write a selector value to **CoE object 0x6000:0x14**. After the selector is written, the selected measurement becomes available through **CoE object 0x6000:0x1D**, which is read as an **AI32** value.

The following tag attributes support this mechanism:

- **Selector**
  The selector value that must be written before reading the CoE object.
- **SelectorWriteIndex**
  The CoE **Index** used for writing the selector (e.g., 0x6000 for EL3403).
- **SelectorWriteSubIndex**
  The CoE **SubIndex** used for writing the selector (e.g., 0x14 for EL3403).
- **SelectorType**
  Defines the data size of the selector write operation (e.g., 1-byte, 2-byte, or 4-byte write). 1 = Byte , 2 = int16 , 3 = int32
  For EL3403, the selector is a **32-bit integer**, so SelectorType = 3.

This mechanism allows pbsSoftLogic to support multiplexed CoE objects cleanly and efficiently, enabling access to a wide range of measurement values using a single CoE read register.

6000:0	PM Inputs Ch.1	Largest subindex of this object				UINT8
6000:0E	Sync Error	reserved				BOOLEAN
6000:10	TxPDO Toggle	The TxPDO toggle is toggled by the slave when the data of the associated TxPDO is updated.				BOOLEAN
6000:11	Current	Current channel 1	**Unit:** 0.000001 A			INT32
6000:12	Voltage	Voltage channel 1	**Unit:** 0.0001 V			INT32
6000:13	Active power	Active power channel 1	**Unit:** 0.01 W			INT32
6000:14	Index	Acknowledge for variable output value channel 1	**Index (dec)**	**Name**	**Unit**	UINT8
			0	Apparent power	0.01 VA	
			1	Reactive power	0.01 VAR	
			2	Energy	0.001 Wh	
			3	cosPhi	0.001	
			4	Frequency	0.1 Hz	
			5	Energy (negative)	0.001 Wh	
			6-99	reserved	-	
			100	Timestamp of the Distributed Clocks [▶ 137]	1 ns	
			101-255	reserved	-	
6000:1D	VariantValue	variable output value channel 1 (see index 0x6000:14)				INT32

Figure -5

Config

The **Config** attribute is used for CoE objects that must be written during device initialization. When **Config = "1"**, the EtherCAT Master Driver writes the specified value to the CoE object at **startup** and again after every **reconnect**. This is typically used for device parameters, scaling values, filter settings, operating modes, and any configuration data that must be restored before normal operation begins.

When **Config = "0"**, the CoE object is treated as a **runtime value**, and the driver performs continuous read operations for to the object.

CoEScanTime

The **CoEScanTime** attribute defines the individual read interval for each CoE tag. This allows different CoE objects to be polled at different rates based on their importance or update frequency. For example, if frequency needs to be read every **10 seconds**, and active power every **5 seconds**, you can set CoEScanTime = "10000" for the frequency tag and CoEScanTime = "5000" for the active power tag. Each CoE tag operates independently, ensuring that slow-rate parameters do not affect the performance of faster CoE reads or the real-time PDO cycle.

In this section, we provide several example tag definitions for different EtherCAT devices and demonstrate how EtherCAT Tags are used in practical applications. These examples illustrate how to configure PDO and CoE tags, how to work with selector-based CoE objects, and how to apply the various tag attributes described in the previous chapters. By reviewing these samples, users can quickly understand how to map real EtherCAT device data into pbsSoftLogic and how to structure tags for both simple I/O modules and advanced measurement terminals.

The **Beckhoff EL1008** is an 8-channel digital input terminal that provides fast, cyclic acquisition of boolean input signals. The module operates entirely through **PDOs**, exposing one byte of input data where each bit represents the state of a digital input channel. Because the EL1008 is a simple digital input device with no configurable parameters, it does **not** provide meaningful CoE objects for runtime access. Its Object Dictionary contains only basic identification entries and mandatory EtherCAT information, none of which require or support CoE-based reading or writing during normal operation. As a result, all EL1008 data exchange in pbsSoftLogic is performed exclusively through **PDO tags**, making configuration straightforward and highly deterministic.

The sample tags shown below represent the eight digital input channels of a **Beckhoff EL1008** module assigned to **Slot 2** in the EtherCAT network. Since the EL1008 exposes all of its input data exclusively through **PDOs**, each channel is mapped to a single bit within the same PDO byte. In this configuration, the module's input byte is located at **Byte = 0**, and each digital input channel corresponds to a specific **bit position** defined by the **Index** attribute.

Because the EL1008 has no selector mechanism and no configurable CoE parameters, all CoE-related attributes are set to zero (CoE="0", Selector="0", Config="0", etc.). Each tag is therefore a simple, deterministic PDO digital input that the EtherCAT Master reads every scan cycle.

tag definitions:

<Tag Name="DI2_0" Type="DI" Init="0" Slot="2" Byte="0" Index="0" CoE="0" Selector="0" SelectorWriteIndex="0" SelectorWriteSubIndex="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DI2_1" Type="DI" Init="0" Slot="2" Byte="0" Index="1" CoE="0" Selector="0" SelectorWriteIndex="0" SelectorWriteSubIndex="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DI2_2" Type="DI" Init="0" Slot="2" Byte="0" Index="2" CoE="0" Selector="0" SelectorWriteIndex="0" SelectorWriteSubIndex="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DI2_7" Type="DI" Init="0" Slot="2" Byte="0" Index="7" CoE="0" Selector="0" SelectorWriteIndex="0" SelectorWriteSubIndex="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

These entries correctly map channels **0, 1, 2, and 7** of the EL1008 module. Channels 3–6 would follow the same pattern with Index="3" through Index="6".

The **Beckhoff EL2024** is a 4-channel digital output terminal designed for switching 24 V DC loads with a maximum current of **2 A per channel**. Like other EL2xxx digital output modules, the EL2024 operates entirely through **PDOs**, exposing one output byte where each bit corresponds to a digital output channel. The module does not provide any meaningful CoE objects for runtime access or configuration; its CoE dictionary contains only standard identification entries required by EtherCAT, with no adjustable parameters. As a result, all interaction with the EL2024 in pbsSoftLogic is performed through **PDO write tags**, making configuration straightforward and ensuring deterministic output updates during each EtherCAT scan cycle.

<Tag Name="DO3_0" Type="DO" Init="0" Slot="3" Byte="0" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DO3_1" Type="DO" Init="0" Slot="3" Byte="0" Index="1" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DO3_2" Type="DO" Init="0" Slot="3" Byte="0" Index="2" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DO3_3" Type="DO" Init="0" Slot="3" Byte="0" Index="3" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

The **Beckhoff EL3632** is a 2-channel IEPE/ICP vibration and accelerometer input terminal designed for high-resolution dynamic signal acquisition. Each channel provides a **32-bit signed measurement value**, a **32-bit timestamp/sample counter**, and a **1-byte status field**, resulting in **9 bytes per channel** and a total PDO size of **18 bytes**. These values are delivered exclusively through **input PDOs**, ensuring deterministic, high-speed sampling suitable for vibration monitoring, condition analysis, and dynamic measurement applications. The EL3632 also exposes several **CoE objects**, but these are primarily used for configuration—such as excitation current settings, measurement ranges, and diagnostic parameters—not for cyclic data acquisition.

<Tag Name="EL3632_V1" Type="AI32" Init="0" Slot="9" Byte="0" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="EL3632_TS1" Type="AI32" Init="0" Slot="9" Byte="4" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="EL3632_S1" Type="BI" Init="0" Slot="9" Byte="8" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="EL3632_V2" Type="AI32" Init="0" Slot="9" Byte="9" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="EL3632_TS2" Type="AI32" Init="0" Slot="9" Byte="13" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="EL3632_S2" Type="BI" Init="0" Slot="9" Byte="17" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

Following tag is read Vendor ID every 5 sec from EL3630 .

<Tag Name="VendorID9" Type="AI32" Init="0" Slot="9" Byte="0x1018" Index="0x01" CoE="1" Selector="-1" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="5000" Log="0" />

The **Beckhoff EL3403** is a 3-phase electrical power measurement terminal designed for monitoring voltage, current, power, and energy parameters in AC systems. Unlike simple I/O terminals, the EL3403 uses a **VariantValue PDO structure**, where the cyclic PDO provides a set of core measurement values—typically phase voltages, phase currents, and active power—encoded as **32-bit signed integers**. More advanced parameters such as reactive power, apparent power, frequency, power factor, and energy counters are not directly available in PDO form. Instead, the EL3403 exposes a rich set of **CoE objects** under index **0x6000**, many of which are **multiplexed**. To access these values, the EtherCAT Master must first write a **selector** to CoE object **0x6000:0x14**, after which the selected measurement becomes available at **0x6000:0x1D** as a 32-bit value. This makes the EL3403 a hybrid device: PDOs provide fast, deterministic real-time data, while CoE access enables retrieval of extended measurement parameters. In pbsSoftLogic, the EL3403 is handled using a combination of **AI32 PDO tags** and **selector-based CoE tags**, ensuring full access to all measurement capabilities of the terminal.

<Tag Name="IL1" Type="AI32" Init="0" Slot="5" Byte="2" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="VL1" Type="AI32" Init="0" Slot="5" Byte="6" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="P1" Type="AI32" Init="0" Slot="5" Byte="10" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Index1" Type="BI" Init="0" Slot="5" Byte="14" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Var1" Type="AI32" Init="0" Slot="5" Byte="15" Index="0" CoE="0" Selector="0" SelectorWriteIndex ="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

Status Tags — EtherCAT Master and Slot State Monitoring

pbsSoftLogic allows the definition of **Status Tags** for both the EtherCAT Master and any individual Slot by assigning the tag **Type = "SYS"** and specifying the corresponding **Slot** number. A Status Tag provides a real-time indication of the operational state of the EtherCAT system. For the **Master**, the status value becomes **1** when **more than 50% of all detected slots are in OPERATIONAL state**, indicating that the network is functioning correctly. If any initialization error occurs, or if **50% or more of the slots fall out of OPERATIONAL**, the Master status automatically changes to **0**, signaling a system-level fault. For **individual Slots**, the Status Tag reflects the EtherCAT state machine value reported by the slave, using the standard EtherCAT state codes: 0x00 (NONE), 0x01 (INIT), 0x02 (PRE-OP), 0x03 (BOOT), 0x04 (SAFE-OP), 0x08 (OPERATIONAL), and 0x10 (ERROR/ACK). These status values allow pbsSoftLogic to monitor the health of each EtherCAT device and the overall network, enabling deterministic fault detection and safe system behavior.

<Tag Name="Status" Type="SYS" Init="0" Slot="0" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Status5" Type="SYS" Init="0" Slot="5" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Status10" Type="SYS" Init="0" Slot="10" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Status3" Type="SYS" Init="0" Slot="3" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Status2" Type="SYS" Init="0" Slot="2" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="Status9" Type="SYS" Init="0" Slot="9" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

Figure 6 illustrates how pbsSoftLogic reads and reports the operational status of each EtherCAT device in the network. In this example, the system contains several Beckhoff terminals, each of which can expose a dedicated **Status Tag** by defining the tag with **Type = "SYS"** and assigning the corresponding **Slot** number.

Found 7 EtherCAT slaves:

Slot 1 : EK1100 (Inputs=0 bytes, Outputs=0 bytes)

Slot 2 : EL1008 (Inputs=1 bytes, Outputs=0 bytes)

Slot 3 : EL2024 (Inputs=0 bytes, Outputs=0 bytes)

Slot 4 : EL3403 (Inputs=62 bytes, Outputs=3 bytes)

Slot 5 : EL3054 (Inputs=16 bytes, Outputs=0 bytes)

Slot 6 : EL3122 (Inputs=6 bytes, Outputs=0 bytes)

Slot 7 : EL4122 (Inputs=0 bytes, Outputs=4 bytes)

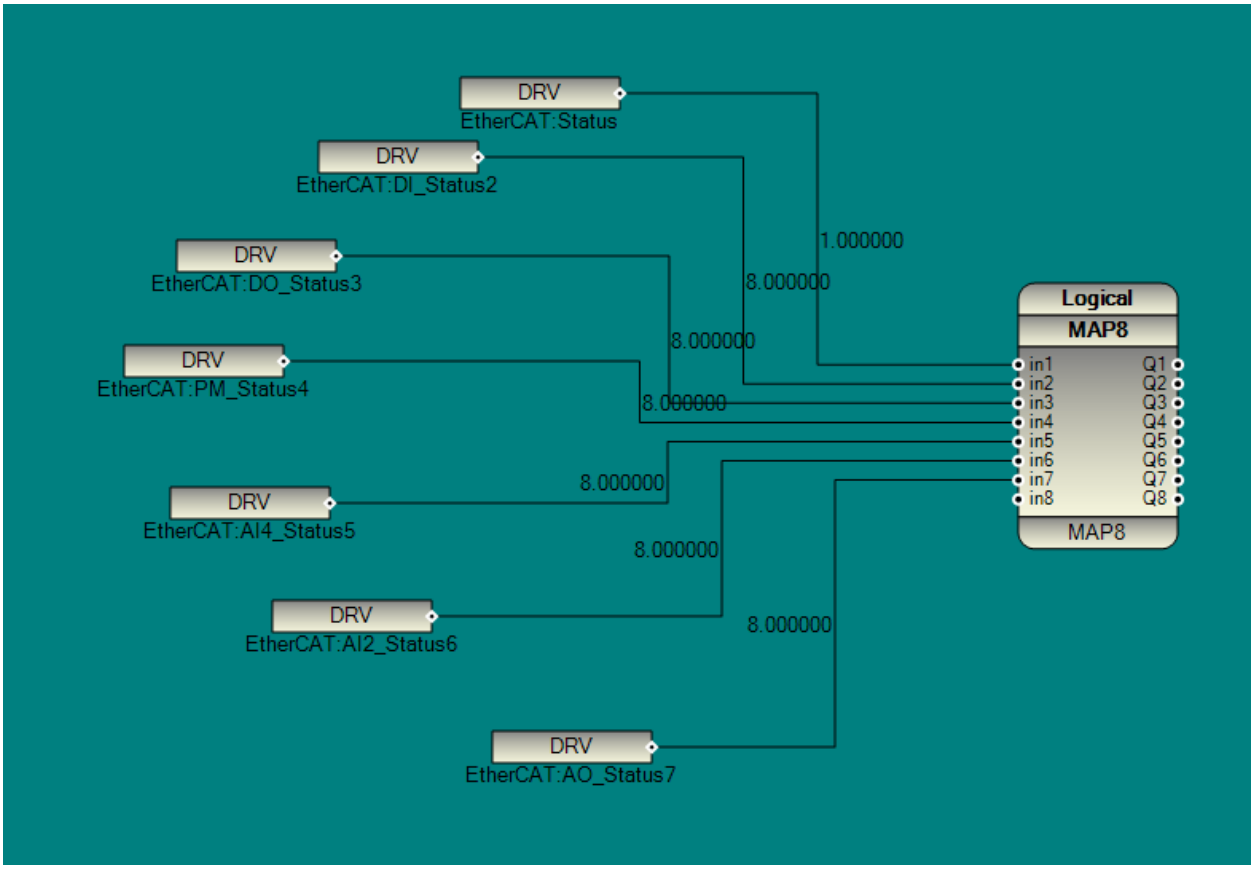


Figure -6

And we define following tags for EtherCAT Driver .

<Tag Name="Status" Type="SYS" Init="0" Slot="0" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DI_Status2" Type="SYS" Init="0" Slot="2" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="DO_Status3" Type="SYS" Init="0" Slot="3" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="PM_Status4" Type="SYS" Init="0" Slot="4" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="AI4_Status5" Type="SYS" Init="0" Slot="5" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="AI2_Status6" Type="SYS" Init="0" Slot="6" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />

<Tag Name="AO_Status7" Type="SYS" Init="0" Slot="7" Byte="0" CoE="0" Selector="0" SelectorWriteIndex="0" Index="0" SelectorWriteSubIndex ="0" Config="0" SelectorType="0" CoEScanTime="0" Log="0" />