



# User Manual

# EtherCAT Library API

86Duino Coding IDE 501

EtherCAT Library

(Version3.3)

## REVISION

Date	Version	Description
2023/5/5	Version1.0	Initial release.
2023/10/24	Version2.0	Updated API.
2024/8/1	Version2.2	Updated API.
2025/3/20	Version3.0	<ul style="list-style-type: none"> <li>Updated API information for 86Duino IDE 501.</li> <li>Change master to MDevice.</li> <li>Change Slave to SubDevice.</li> <li>Add <code>errGetCableBrokenLocation1()</code> figure.</li> <li>Adjust The Encoder Reading Object of The Three-Axis Stepper Motor Controller SubDevice (Affects <code>encoder.read()</code> function):               <ol style="list-style-type: none"> <li>60e4.0 ==&gt; 502a.1</li> <li>68e4.0 ==&gt; 502a.2</li> <li>70e4.0 ==&gt; 502a.3</li> </ol> </li> <li>Added functions for 3-axis stepper SubDevice.</li> </ul>
2025/4/18	Version3.1	<ul style="list-style-type: none"> <li>Corrected typos in multiple API comments (Pages 34, 89, 162, 176, 182, 248–280)</li> <li>Fixed duplication and grammar in function descriptions.</li> <li>Updated UART-related notes for CIO Series.</li> <li>Confirmed <code>isSupportDC()</code> applies to SubDevice.</li> <li>Fixed incorrect type: <code>int writeSII16</code> to <code>int writeSII32</code>.</li> <li>Renamed CIO series model from QEC-R11DFFG to QEC-R11CFFG</li> </ul>
2025/4/29	Version3.2	<ul style="list-style-type: none"> <li>Merged duplicate descriptions of common API functions (e.g., <code>attach()</code>, <code>detach()</code>) into shared references.</li> </ul>
2026/1/7	Version3.3	<ul style="list-style-type: none"> <li>Update Feature Table: “Access to EEPROM” and “Boot State” functions.</li> <li>Update all example code: “slave” to “device”.</li> <li>Add <code>EthercatDevice_QECRXXDXXK</code> series DIO.</li> <li>Add <code>EthercatDevice_DmpAIQ_Generic</code> derived class items.</li> <li>Add <code>EthercatDevice_DmpCIQ_Generic</code> derived class items.</li> <li>Delete <code>EthercatDevice_DmpLCD_Generic</code> class.</li> <li>Add <code>EthercatDevice_DmpStepper_Generic</code> derived class items.</li> </ul>

- |  |  |  |
|--|--|--|
|  |  | <ul style="list-style-type: none"><li>• Update SubDevice Information Examples object name.</li></ul> |
|--|--|--|

## COPYRIGHT

The information in this manual is subject to change without notice for continuous improvement in the product. All rights are reserved. The manufacturer assumes no responsibility for any inaccuracies that may be contained in this document and makes no commitment to update or to keep current the information contained in this manual.

No part of this manual may be reproduced, copied, translated or transmitted, in whole or in part, in any form or by any means without the prior written permission of the ICOP Technology Inc.

©Copyright 2026 ICOP Technology Inc.

Ver.3.3 January, 2026

## TRADEMARKS ACKNOWLEDGMENT

ICOP® is the registered trademark of ICOP Corporation. Other brand names or product names appearing in this document are the properties and registered trademarks of their respective owners. All names mentioned herewith are served for identification purpose only.

For more detailed information or if you are interested in other ICOP products, please visit our official websites at:

- Global: [www.icop.com.tw](http://www.icop.com.tw)
- USA: [www.icoptech.com](http://www.icoptech.com)
- Japan: [www.icop.co.jp](http://www.icop.co.jp)
- Europe: [www.icoptech.eu](http://www.icoptech.eu)
- China: [www.icop.com.cn](http://www.icop.com.cn)

For technical support or drivers download, please visit our websites at:

- [https://www.icop.com.tw/resource\\_entrance](https://www.icop.com.tw/resource_entrance)

For EtherCAT solution service, support or tutorials, 86Duino Coding IDE 500+ introduction, functions, languages, libraries, etc. Please visit the QEC website:

- QEC: <https://www.qec.tw/>

This Manual is for the QEC series.

## SAFETY INFORMATION

- Read these safety instructions carefully.
- Please carry the unit with both hands and handle it with caution.
- Power Input voltage +19 to +50VDC Power Input (Typ. +24VDC)
- Make sure the voltage of the power source is appropriate before connecting the equipment to the power outlet.
- To prevent the QEC device from shock or fire hazards, please keep it dry and away from water and humidity.
- Operating temperature between -20 to +70°C/-40 to +85°C (Option).
- When using external storage as the main operating system storage, ensure the device's power is off before connecting and removing it.
- Never touch un-insulated terminals or wire unless your power adaptor is disconnected.
- Locate your QEC device as close as possible to the socket outline for easy access and avoid force caused by the entangling of your arms with surrounding cables from the QEC device.
- If your QEC device will not be used for a period of time, make sure it is disconnected from the power source to avoid transient overvoltage damage.

### WARNING!



DO NOT ATTEMPT TO OPEN OR TO DISASSEMBLE THE CHASSIS (ENCASING) OF THIS PRODUCT. PLEASE CONTACT YOUR DEALER FOR SERVICING FROM QUALIFIED TECHNICIAN.

# CONTENT

Ch. 1	Introduction.....	1
1.1	About QEC EtherCAT Controller .....	3
1.1.1	What is 86Duino IDE? .....	3
1.1.2	Architecture .....	4
1.1.3	Hardware Platform .....	5
1.1.4	Dual-System Synchronization .....	6
1.2	Features .....	7
1.2.1	Feature Table.....	7
1.3	Feature Packs.....	10
1.3.1	Cable Redundancy .....	10
1.3.1.1	Case 1: Without Cable Broken .....	11
1.3.1.2	Case 2: Cable Broken between two SubDevices.....	12
1.3.1.3	Case 3: Cable broken between MDevice and SubDevice .....	13
1.4	Synchronization.....	14
1.4.1	Free Run.....	15
1.4.2	SM-Synchronous.....	16
1.4.3	DC-Synchronous .....	17
1.5	Benchmark.....	19
1.5.1	System Variables .....	19
1.5.2	Measurement Functions .....	20
1.5.3	Measurement Result .....	22
1.5.3.1	Configured Cycle Time equals 125 $\mu$ s .....	22
1.5.3.2	Configured Cycle Time equals 250 $\mu$ s .....	23
1.5.3.3	Configured Cycle Time equals 500 $\mu$ s .....	23
1.5.3.4	Configured Cycle Time equals 1000 $\mu$ s.....	24
1.5.4	Example Code .....	25
1.5.5	EtherCAT MDevice Cyclic Frame Jitter .....	27
Ch. 2	Functions.....	28
2.1	EtherCAT MDevice.....	30
2.1.1	EtherCAT MDevice Settings .....	33
2.1.2	Initialization Functions .....	38
2.1.3	Control Functions.....	46
2.1.4	Callback Functions .....	61
2.1.5	SubDevice Information Functions .....	77
2.2	EtherCAT SubDevice .....	86
2.2.1	_EthercatDevice_CommonDriver .....	87
2.2.1.1	Information Functions .....	90
2.2.1.2	PDO Functions .....	104
2.2.1.3	CoE Functions .....	121
2.2.1.4	FoE Functions .....	144

2.2.1.5	Distributed Clock (DC) Functions .....	149
2.2.1.6	SII EEPROM Functions .....	153
2.2.2	EthercatDevice_Generic .....	163
2.2.2.1	Initialization Functions .....	164
2.2.3	EthercatDevice_CiA402 .....	168
2.3	QEC-Series SubDevice .....	173
2.3.1	_EthercatDevice_DmpCommonDriver .....	175
2.3.1.1	System Monitoring Functions.....	177
2.3.1.2	MTBF Functions.....	193
2.3.1.3	Order Information Functions.....	200
2.3.2	EthercatDevice_DmpDIQ_Generic .....	205
2.3.2.1	Initialization Functions .....	208
2.3.2.2	Digital I/O Functions.....	211
2.3.2.3	Broken Wire Detection Functions .....	217
2.3.3	EthercatDevice_DmpAIQ_Generic .....	220
2.3.3.1	Initialization Functions .....	222
2.3.3.2	Analog Output Access Functions.....	225
2.3.3.3	Analog Input Access Functions.....	232
2.3.4	EthercatDevice_DmpCIQ_Generic .....	235
2.3.4.1	Initialization Functions .....	238
2.3.4.2	Control Functions .....	241
2.3.4.3	Digital I/O Functions.....	243
2.3.4.4	Analog I/O Functions.....	249
2.3.4.5	UART Functions .....	258
2.3.5	EthercatDevice_DmpHID_Generic .....	291
2.3.5.1	Initialization Functions .....	295
2.3.5.2	Control Functions .....	298
2.3.5.3	UART Functions .....	300
2.3.5.4	LCM Functions.....	334
2.3.5.5	Keypad Functions .....	346
2.3.5.6	MPG Functions .....	351
2.3.5.7	Buzzer Functions .....	368
2.3.6	EthercatDevice_DmpStepper_Generic .....	371
2.3.6.1	Initialization Functions .....	375
2.3.6.2	Control Functions .....	380
2.3.6.3	Machine Functions .....	388
2.3.6.4	Encoder Functions.....	419
2.3.6.5	Configuration Functions.....	426
Ch. 3	Examples.....	489
3.1	SubDevice Information Examples.....	490
3.1.1	Example 1: Using EthercatMaster class .....	490
3.1.2	Example 2: Using EthercatDevice_Generic class .....	491

3.1.3	Example 3: Using EthercatDevice_CiA402 class .....	493
3.2	SDO Upload/Download Examples .....	495
3.2.1	Example 1: SDO Upload using sdoUpload8() .....	495
3.2.2	Example 2: SDO Upload using sdoUpload() .....	496
3.2.3	Example 3: SDO Upload using sdoUpload() with abort code .....	497
3.2.4	Example 4: SDO Download using sdoDownload8() .....	498
3.2.5	Example 5: SDO Download using sdoDownload() .....	499
3.2.6	Example 6: SDO Download using sdoDownload() with abort code.....	500
3.2.7	Example 7: Print the PDO mapping configuration.....	501
3.2.8	Example 8: Change the PDO mapping configuration.....	503
3.3	PDO Read/Write Examples.....	505
3.3.1	Example 1: Read a bit data from Input PDO using pdoBitRead() .....	505
3.3.2	Example 2: Read a byte data from Input PDO using pdoRead8() .....	506
3.3.3	Example 3: Read data from Input PDO using pdoRead() .....	507
3.3.4	Example 4: Write a bit data to Output PDO using pdoBitWrite() .....	508
3.3.5	Example 5: Write a byte data to Output PDO using pdoWrite8() .....	509
3.3.6	Example 6: Write data to Output PDO using pdoWrite() .....	510
3.4	Callback Examples.....	511
3.4.1	Example 1: Cyclic callback.....	511
3.4.2	Example 2: Cyclic callback with FPU-enabled .....	513
3.4.3	Example 3: Error callback.....	515
3.4.4	Example 4: Event callback.....	517
3.5	DC Examples .....	518
3.5.1	Example 1: Enable DC synchronization.....	518
Appendix	.....	520
A.1	Error List .....	521
A.2	Error Description and Corrective Actions .....	524
A.3	Error Callback Code.....	547
A.4	Event Callback Code.....	548
A.5	SDO Abort Code .....	549
A.6	Data Type.....	551
A.7	EtherCAT Network Information.....	553
Warranty	.....	556

# Ch. 1

## Introduction

EtherCAT (Ethernet for Control Automation Technology) is a communication protocol designed specifically for industrial control applications. It is a high-performance, real-time communication technology with extremely low communication latency and high bandwidth, suitable for various industrial automation and control systems.



Key features of EtherCAT include:

- **High-speed Real-time Performance:** EtherCAT can transmit data at very high speeds while maintaining extremely low communication latency, enabling high levels of real-time performance suitable for demanding control applications.
- **Flexible Scalability:** EtherCAT networks can easily scale to support large numbers of nodes and complex network topologies, making them suitable for various scales and application scenarios.
- **Open Standard:** EtherCAT is an open standard with widespread industrial support and application. It has rich libraries and tools that can be easily integrated into existing industrial automation systems.
- **Cost-effectiveness:** EtherCAT uses standard Ethernet hardware without the need for additional specialized hardware, reducing costs and deployment complexity.

EtherCAT is commonly used in industrial automation, robotics control, motion control, embedded systems, and real-time data acquisition, providing high-performance and reliable communication solutions for these applications.

## 1.1 About QEC EtherCAT Controller

QEC (Quicker, Easier Control with EtherCAT) is an EtherCAT controller developed by ICOP. It offers high synchronization and real-time capabilities, along with ease of development. Below, we'll describe the architecture, features, and performance of QEC.

QEC MDevice is an EtherCAT MDevice System compatible with 86Duino Coding IDE 500+. It offers real-time EtherCAT communication between EtherCAT MDevice and EtherCAT SubDevices. Except for the EtherCAT Library of 86Duino IDE, QEC MDevice also provides Modbus, Ethernet TCP/IP, CAN bus, etc. industrial communication protocols and uses a rich high-level C/C++ programming language for rapid application development.

### 1.1.1 What is 86Duino IDE?

The 86Duino integrated development environment (IDE) software makes it easy to write and upload code to 86Duino boards and QEC MDevices. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Arduino IDE, Processing, DJGPP, and other open-source software, which can be downloaded from <https://www.qec.tw/software/>.



QEC MDevice's software, 86Duino IDE, also offers a configuration utility: 86EVA, a graphic user interface tool for users to edit parameters for the EtherCAT network; its functions are as follows:

- EtherCAT SubDevices scanning
- Import ENI file
- Setting EtherCAT MDevice
- Configure EtherCAT SubDevices

For other detailed functions, please refer to the [86EVA User Manual](#).

## 1.1.2 Architecture

The EtherCAT MDevice software is primarily divided into two parts, each running on the respective systems of the Vortex86EX2 CPU.

They are responsible for the following tasks:

- **EtherCAT MDevice Library**
  - Provides a C/C++ application interfaces:
    - Initialization interface.
    - Configuration interface.
    - Process Data (PDO) access interface.
    - CAN application protocol over EtherCAT (CoE) access interface.
    - File Access over EtherCAT (FoE) access interface.
    - SubDevice Information Interface (SII) access interface.
    - Distributed Clocks (DC) access interface.
- **EtherCAT MDevice Firmware**
  - Executes the EtherCAT MDevice Core.
  - Controls the Primary/Secondary Ethernet Driver, sending EtherCAT frames.

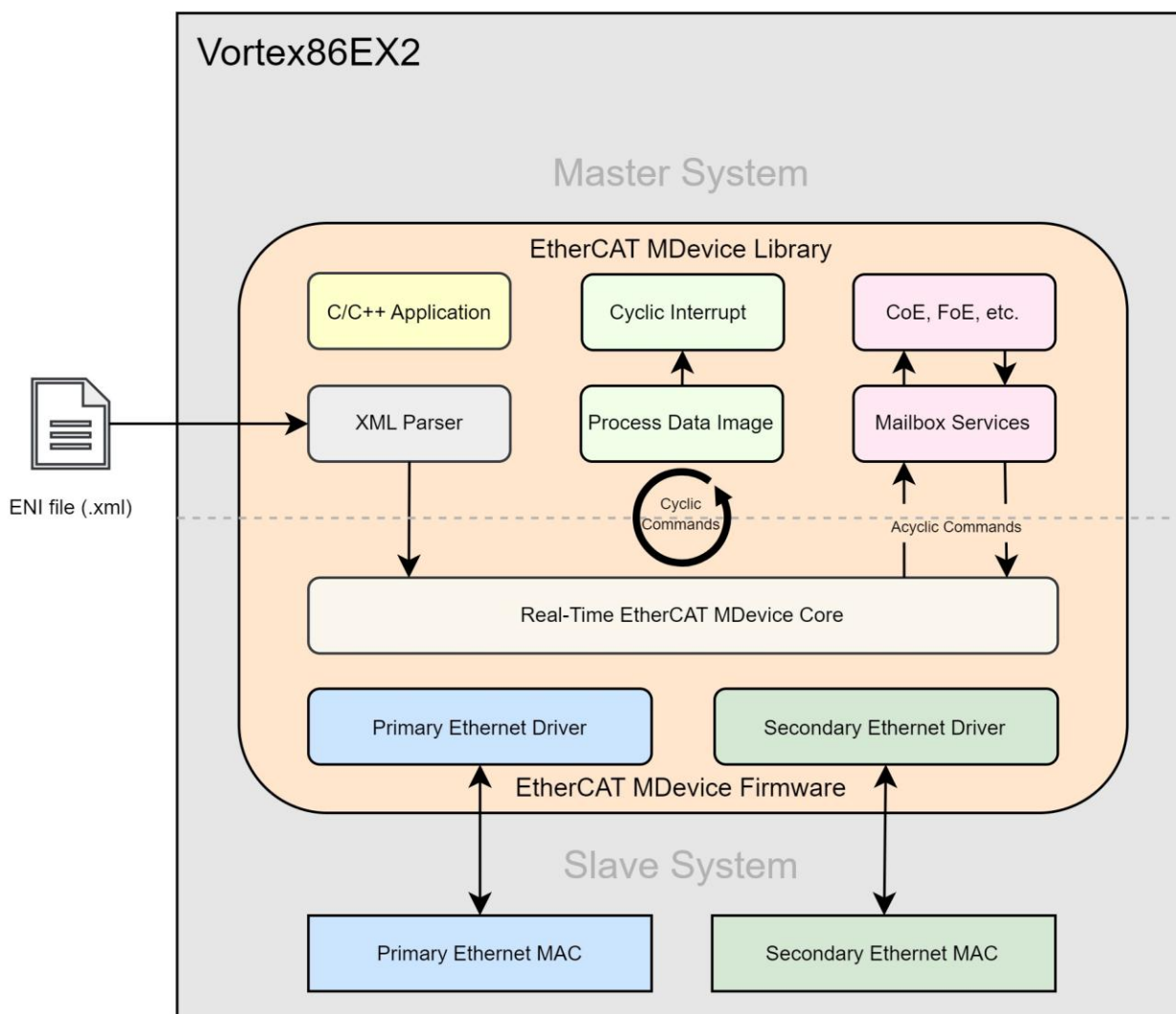
The programs are designed to run on the FreeDOS operating system and have been compiled using the GCC compiler provided by the DJGPP environment.

### 1.1.3 Hardware Platform

The EtherCAT MDevice software only runs on the Vortex86EX2 CPU produced by DM&P, which features a dual-system architecture. It is divided into Master System and Slave System, each running its own operating system, with communication between systems facilitated by Dual-Port RAM and event interrupts. Their respective tasks are as follows:

- **Master System**
  - User's EtherCAT application.
  - User's HMI application.
  - User's Ethernet application.
  - And so on.
- **Slave System**
  - Only responsible for running the EtherCAT MDevice Firmware.

As most applications run on the Master System, the EtherCAT MDevice Firmware running on the Slave System is free from interference by other applications. This setup allows it to focus on executing the EtherCAT MDevice Core, ensuring the synchronization and real-time capabilities of EtherCAT.



## 1.1.4 Dual-System Synchronization

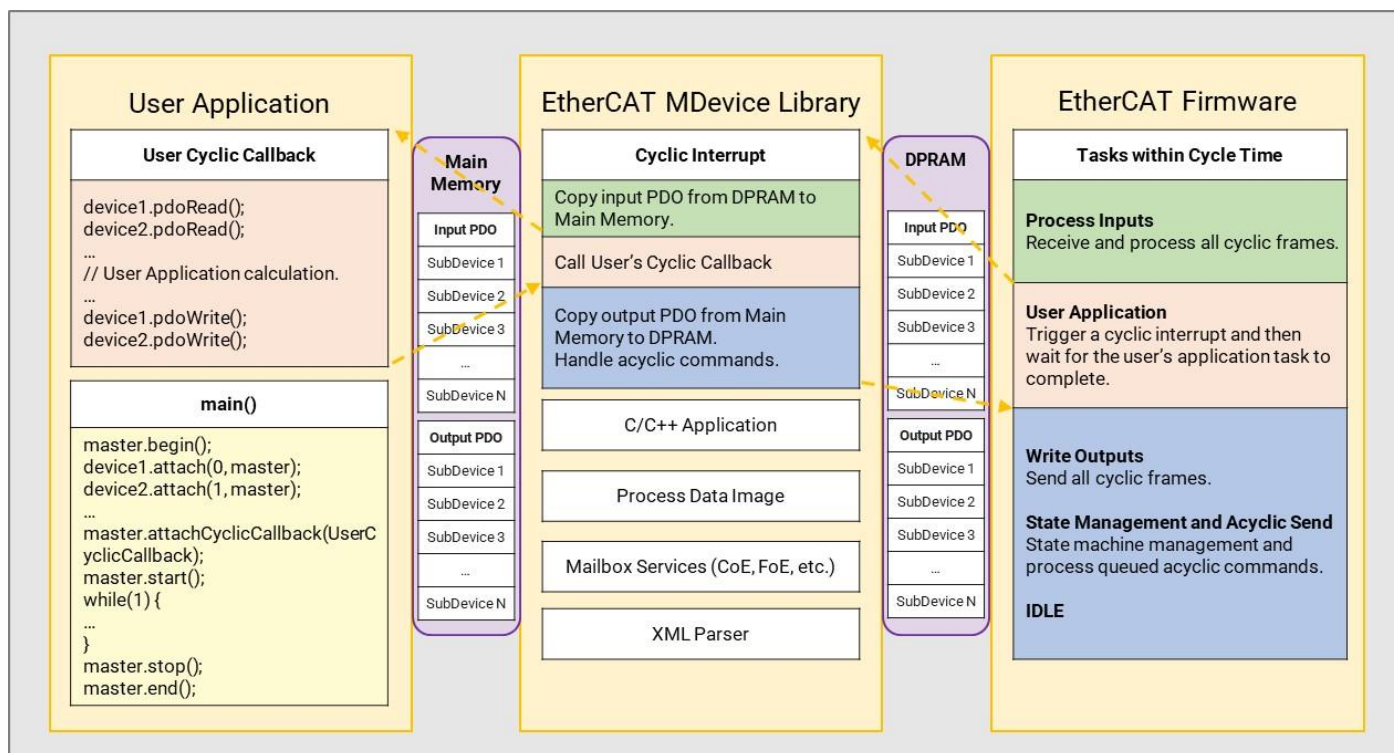
The primary focus of this section is the synchronization of dual-system PDO data. As illustrated in the diagram below, the User Application and EtherCAT MDevice Library blocks run on the Master System, while the Real-Time EtherCAT MDevice Core runs on the Slave System.

When the EtherCAT MDevice Core reaches the **Process Inputs** stage, it receives all cyclic frames from the Ethernet Driver and copies Input PDO data to the DPRAM.

Upon reaching the **User Application** stage, the EtherCAT MDevice Core triggers a Cyclic Interrupt to the Master System. Upon receiving the Cyclic Interrupt, the Master System executes the interrupt handling procedure of the EtherCAT MDevice Library. It moves Input PDO data from DPRAM to Main Memory, calls the user-registered Cyclic Callback, transfers Output PDO data from Main Memory to DPRAM after the Cyclic Callback completes, processes acyclic commands, and concludes the interrupt handling procedure. At this point, both the EtherCAT MDevice Core's **User Application** and the interrupt handling procedure are completed simultaneously.

When the EtherCAT MDevice Core reaches the **Write Outputs** stage, it copies Output PDO data from DPRAM to the Ethernet Driver's DMA and sends frames.

These tasks are executed periodically in a cyclic manner, following the outlined procedural steps, ensuring the synchronization of dual-system PDO data.



## 1.2 Features

The EtherCAT Technology Group defined two classes of EtherCAT MDevice software implementation in [ETG.1500](#). This specification defines MDevice Classes with a well-defined set of MDevice functionalities.

In order to keep things simple only 2 MDevice Classes are defined:

- Class A: Standard EtherCAT MDevice
- Class B: Minimum EtherCAT MDevice

You will see the comparison among Class A, Class B, and our QEC MDevice as follow.

### 1.2.1 Feature Table

Word Usage:

- **shall** equals is required to.
- **should** equals is recommended that.
- **may** equals is permitted to.
- **O** equals supported.

Feature Name	Short Description	Class A	Class B	QEC MDevice
<b>Basic Features</b>				
Service Commands	Support of all commands	shall if ENI import support		O
IRQ field in datagram	Use IRQ information from SubDevice in datagram header	should	should	--
SubDevices with Device Emulation	Support SubDevices with and without application controller	shall	shall	O
EtherCAT State Machine	Support of ESM special behavior	shall	shall	O
Error Handling	Checking of network or SubDevice errors, e.g. Working Counter	shall	shall	O
VLAN	Support VLAN Tagging	may	may	--
EtherCAT Frame Types	Support EtherCAT Frames	shall	shall	O
UDP Frame Types	Support UDP Frames	may	may	--

Feature Name	Short Description	Class A	Class B	QEC MDevice
<b>Process Data Exchange</b>				
Cyclic PDO	Cyclic process data exchange	shall	shall	0
Multiple Tasks	Different cycle tasks. Multiple update rates for PDO	may	may	--
Frame repetition	Send cyclic frames multiple times to increase immunity	may	may	--
<b>Network Configuration</b>				
Online scanning	Network configuration functionality included in EtherCAT MDevice	at least one of them		0
Reading ENI	Network Configuration taken from ENI file	at least one of them		0
Compare Network configuration	Compare configured and existing network configuration during boot-up	shall	shall	0
Explicit Device Identification	Identification used for Hot Connect and prevention against cable swapping	should	should	0
Station Alias Addressing	Support configured station alias in SubDevice	may	may	--
Access to EEPROM	Support routines to access EEPROM via ESC register	Read shall / Write may		0
<b>Mailbox Support</b>				
Support Mailbox	Main functionality for mailbox transfer	shall	shall	0
Mailbox Resilient Layer	Support underlying resilient layer	shall	shall	0
Multiple Mailbox channels	Simultaneous Mailbox protocol transfer to one device	may	may	--
Mailbox polling	Polling Mailbox state in SubDevices	shall	shall	--
<b>CAN application layer over EtherCAT (CoE)</b>				
SDO Up/Download	Normal and expedited transfer	shall	shall	0
Segmented Transfer	Segmented transfer	shall	should	0
Complete Access	Transfer the entire object (with all sub-indices) at once	shall	should (shall if ENI Import supported)	0
SDO Info service	Services to read object dictionary	shall	should	0
Emergency Message	Receive Emergency messages	shall	shall	--
PDO in CoE	PDO services transmitted via CoE	may	may	--

Feature Name	Short Description	Class A	Class B	QEC MDevice
<b>EoE</b>				
EoE protocol	Services for tunneling Ethernet frames. includes all specified EoE services	shall	shall if EoE support	--
Virtual Switch	Virtual Switch functionality	shall	shall if EoE support	--
EoE Endpoint to Operation Systems	Interface to the Operation System on top of the EoE layer	should	should if EoE support	--
<b>FoE</b>				
FoE Protocol	Support FoE Protocol	shall	shall if FoE support	0
Firmware Up/Download	Password, FileName should be given by the application	shall	should	0
Boot State	Support Boot-State for Firmware Up/Download	shall	shall if FW UP/Download	0
<b>SoE</b>				
SoE Services	Support SoE Services	shall	shall if SoE support	--
<b>AoE</b>				
AoE Protocol	Support AoE Protocol	should	should	--
<b>VoE</b>				
VoE Protocol	External Connectivity supported	may	may	--
<b>Synchronization with Distributed Clock (DC)</b>				
DC support	Support of Distributed Clock	shall	shall if DC support	0
Continuous Propagation Delay compensation	Continuous Calculation of the propagation delay	should	should	--
Sync window monitoring	Continuous monitoring of the Synchronization difference in the SubDevices	should	should	--
<b>SubDevice-to-SubDevice Communication</b>				
via MDevice	Information is given in ENI file or can be part of any other network configuration Copying of the data can be handled by MDevice stack or MDevice's application	shall	shall	--
<b>MDevice information</b>				
MDevice Object Dictionary	Support of MDevice Object Dictionary (ETG.5001 MDP sub profile 1100)	should	may	--

## 1.3 Feature Packs

### 1.3.1 Cable Redundancy

EtherCAT Cable Redundancy refers to the capability of the EtherCAT communication system to maintain continuous and reliable communication even in the event of a cable failure. Cable Redundancy employs a ring topology, which is operated in both directions. If one cable fails or is disconnected, another cable path still works to ensure uninterrupted communication.

Cable Redundancy enhances the fault tolerance of the EtherCAT network, minimizing downtime and improving overall system reliability.

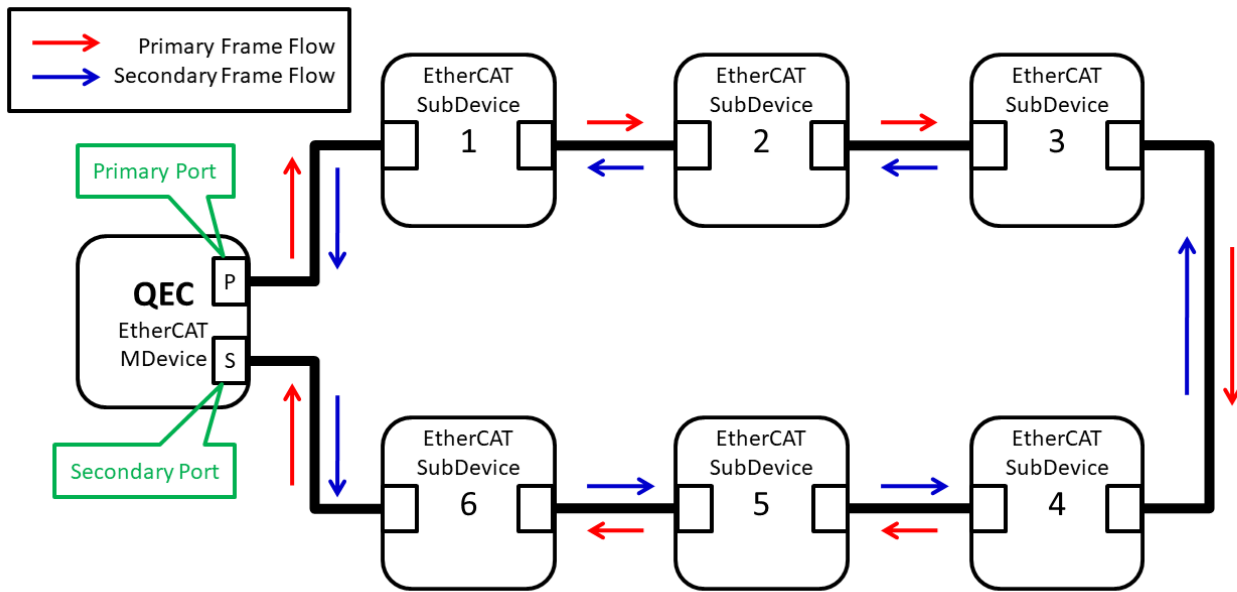
Three scenarios are listed as below regarding whether the cable is broken or not in Cable Redundancy. The following you will see how to work for Cable Redundancy, and the differences for the EtherCAT MDevice Controller between these scenarios.

- Without Cable Broken
- Cable Broken between two SubDevices
- Cable broken between MDevice and SubDevice

For ease of explanation, some assumptions will be made here:

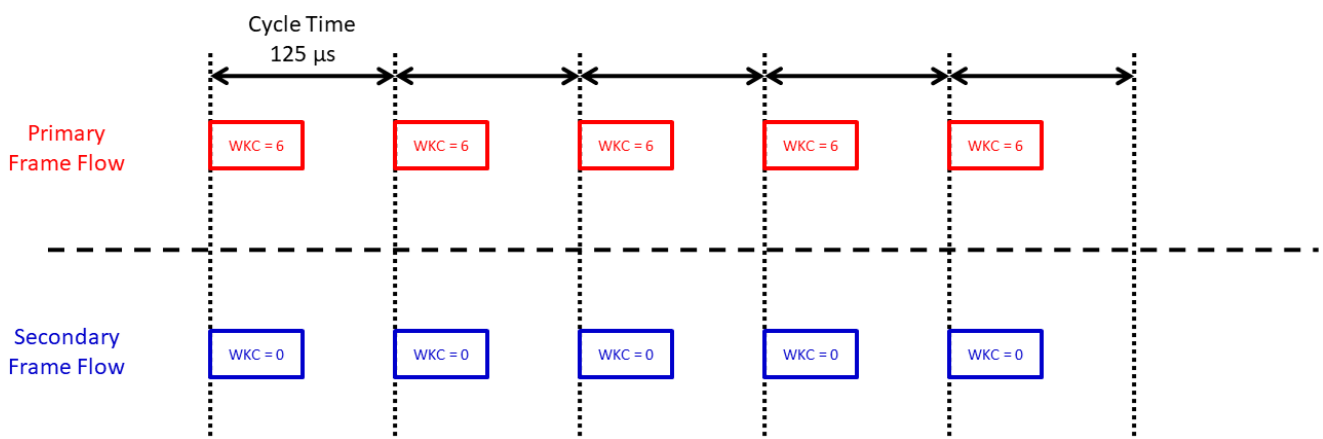
- Assume Cycle Time is set to 125  $\mu$ s.
- Assume all SubDevices only with input PDO (without output PDO), the working counter (WKC) in process data frame will increase 1 when passing through every SubDevices.
- Assume only have 6 SubDevices on EtherCAT network, and the expected working counter (EWKC) is 6.
- Primary Port and Secondary Port send process data frame in every cycle at the same time.

### 1.3.1.1 Case 1: Without Cable Broken



#### Without Cable Broken

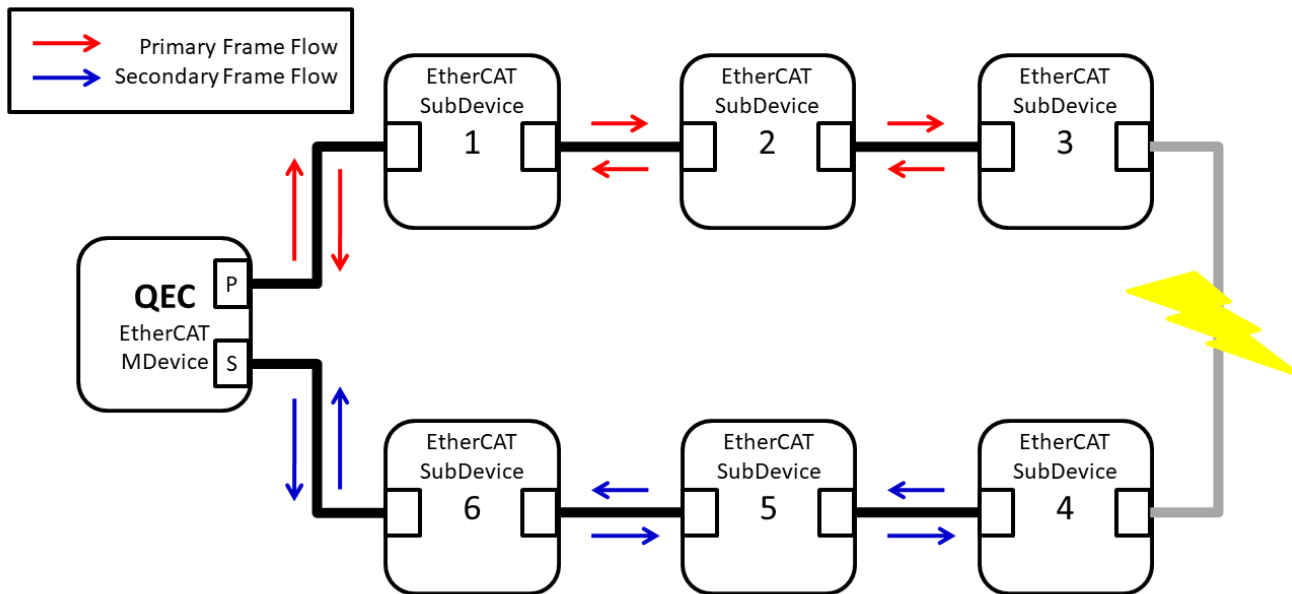
- Primary frame from primary port will be received by secondary port, which frame with WKC = 6.
- Secondary frame from secondary port will be received by primary port, which frame with WKC = 0.
- MDevice will discard the secondary frame because primary frame's WKC equals to EWKC and secondary frame's WKC equals to 0, it means that without cable broken.



### 1.3.1.2 Case 2: Cable Broken between two SubDevices

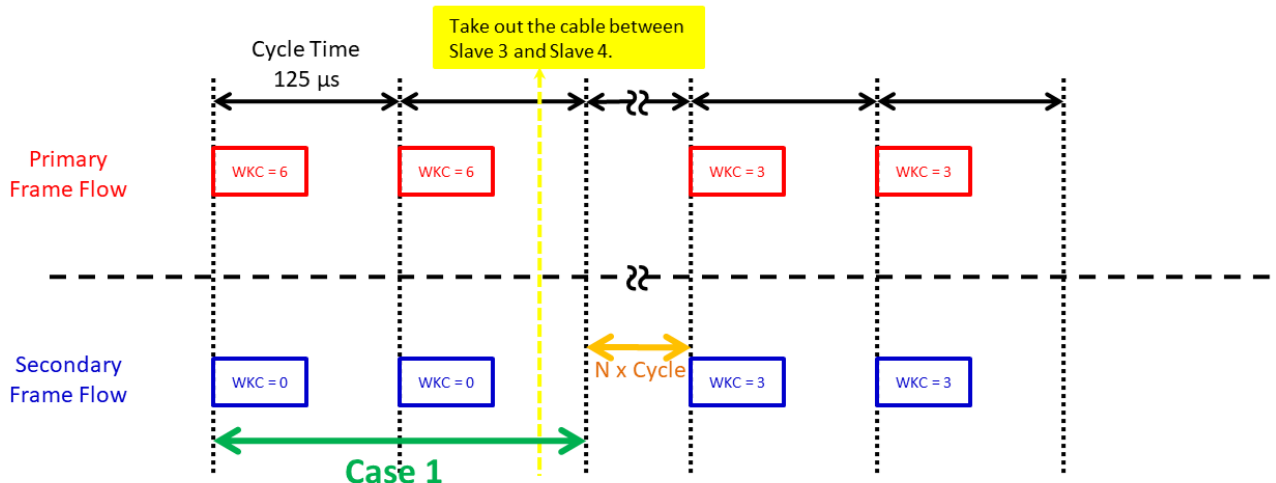
#### Unplug the cable between SubDevice 3 and SubDevice 4

- If you unplug the cable manually, it might with some interference which last several cycles. (N = 0 ~ more)
- If the interference continues for a period of time, some SubDevices will enter the SAFEOP state due to SyncManager Watchdog.



#### Cable broken between SubDevice 3 and SubDevice 4

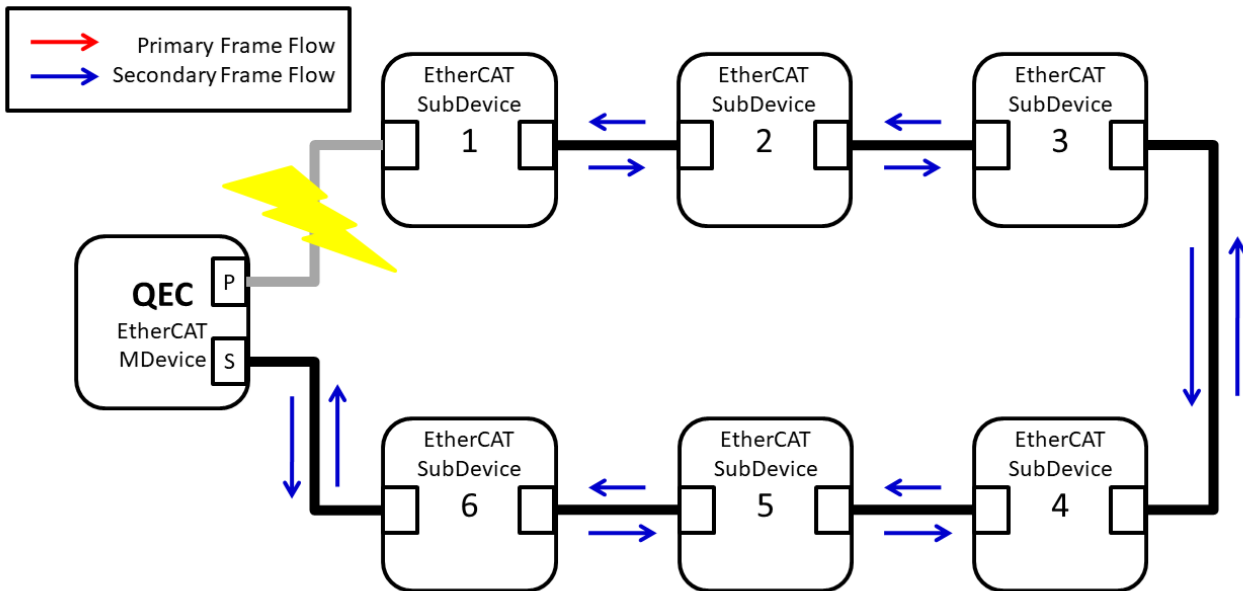
- Primary frame from primary port will be received by primary port, which frame with WKC = 3.
- Secondary frame from secondary port will be received by secondary port, which frame with WKC = 3.
- Primary frame and secondary frame will be combined by MDevice, because primary frame's WKC is less than EWKC and secondary frame's WKC is greater than 0, it means that the cable broken event is happened between two SubDevices.



### 1.3.1.3 Case 3: Cable broken between MDevice and SubDevice

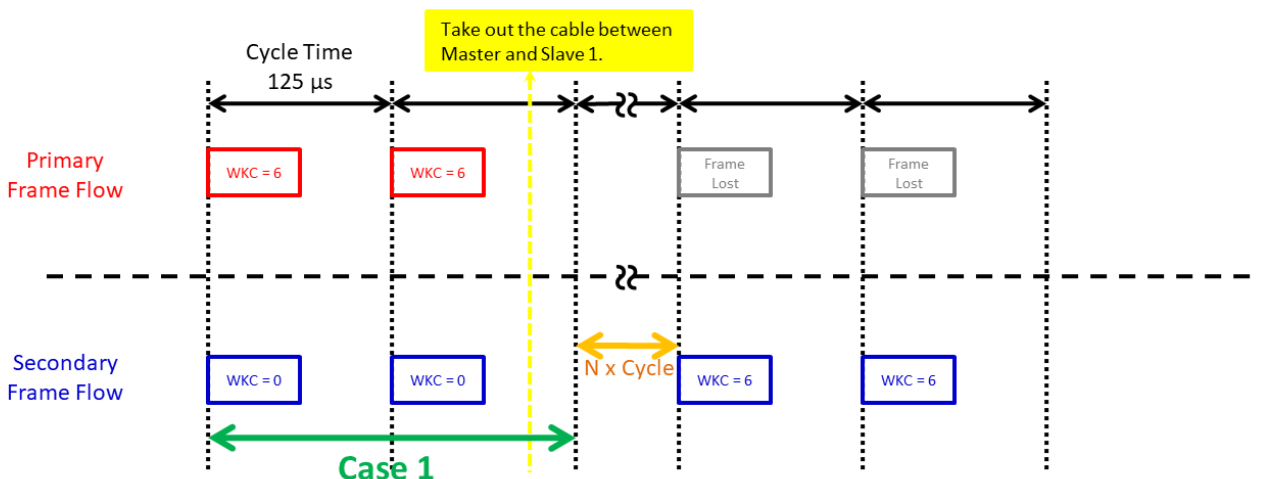
#### Unplug the cable between MDevice and SubDevice 1

- If you unplug the cable manually, it might with some interference which last several cycles. (N = 0 ~ more)
- If the interference continues for a period of time, some SubDevices will enter the SAFEOP state due to SyncManager Watchdog.



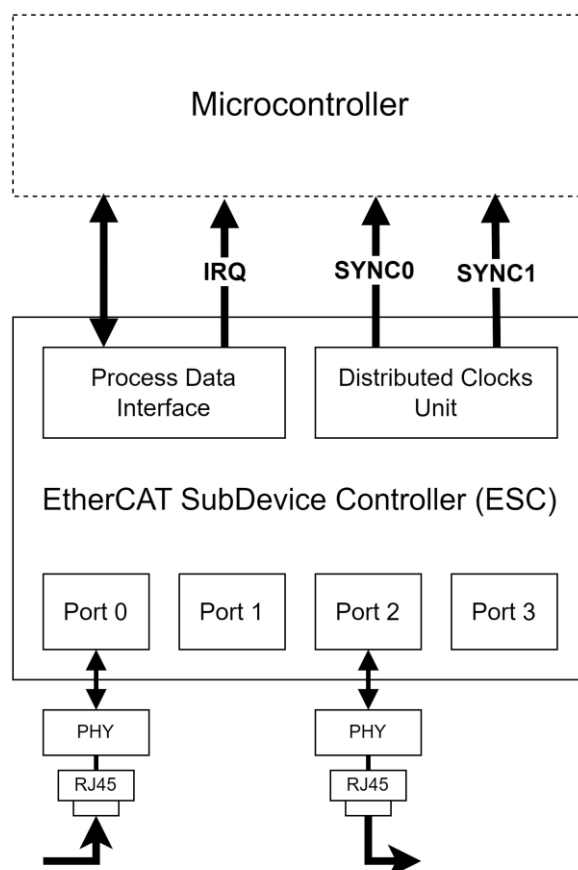
#### Cable broken between MDevice and SubDevice 1

- Primary frame from primary port will be lost.
- Secondary frame from secondary port will be received by secondary port, which frame with WKC = 6.
- MDevice will ignore the primary frame because primary frame was lost and secondary frame's WKC equals to EWKC, it means that the cable broken event is happened between MDevice and SubDevice 1.



## 1.4 Synchronization

The time synchronization among all SubDevices in an EtherCAT network relies on the Distributed Clocks (DC) unit within the EtherCAT SubDevice Controller (ESC), ensuring consistency across the entire system. Typically, the first SubDevice with DC serves as the system reference clock to synchronize other SubDevices with DC. For a more detailed explanation of DC, please refer to [Distributed Clocks](#).



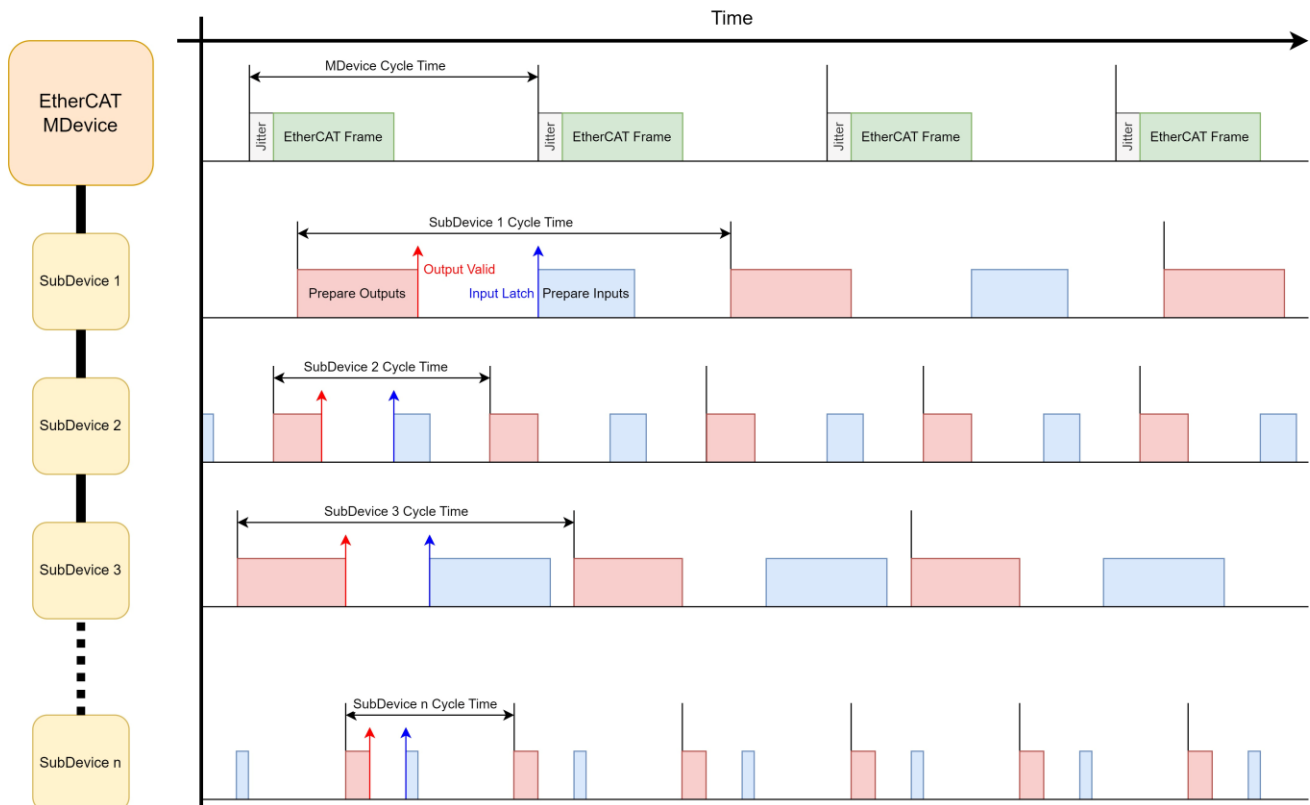
The ESC has three synchronization output pins: **IRQ**, **SYNC0**, and **SYNC1**. The IRQ pin generates a signal to the upper-layer microcontroller ( $\mu$ C) after the ESC receives EtherCAT Cyclic Frames. SYNC0 and SYNC1 pins cyclically generate signals to the  $\mu$ C based on the configuration in the DC related registers of ESC. Hence, if an EtherCAT SubDevice does not have a  $\mu$ C, it does not support synchronization functionality.

There are three synchronization modes in EtherCAT:

- [Free Run](#)
- [SM-Synchronous](#)
- [DC-Synchronous](#)

## 1.4.1 Free Run

The EtherCAT MDevice and all EtherCAT SubDevices each have their own local timer, and their cycle times are independent, so they are not synchronized. As shown in the diagram below, both the EtherCAT MDevice and SubDevice 1, SubDevice 2, SubDevice 3 to SubDevice n have their own Cycle Time, resulting in inconsistent **Output Valid** and **Input Latch**. This scenario is not suitable for applications with high synchronization requirements.

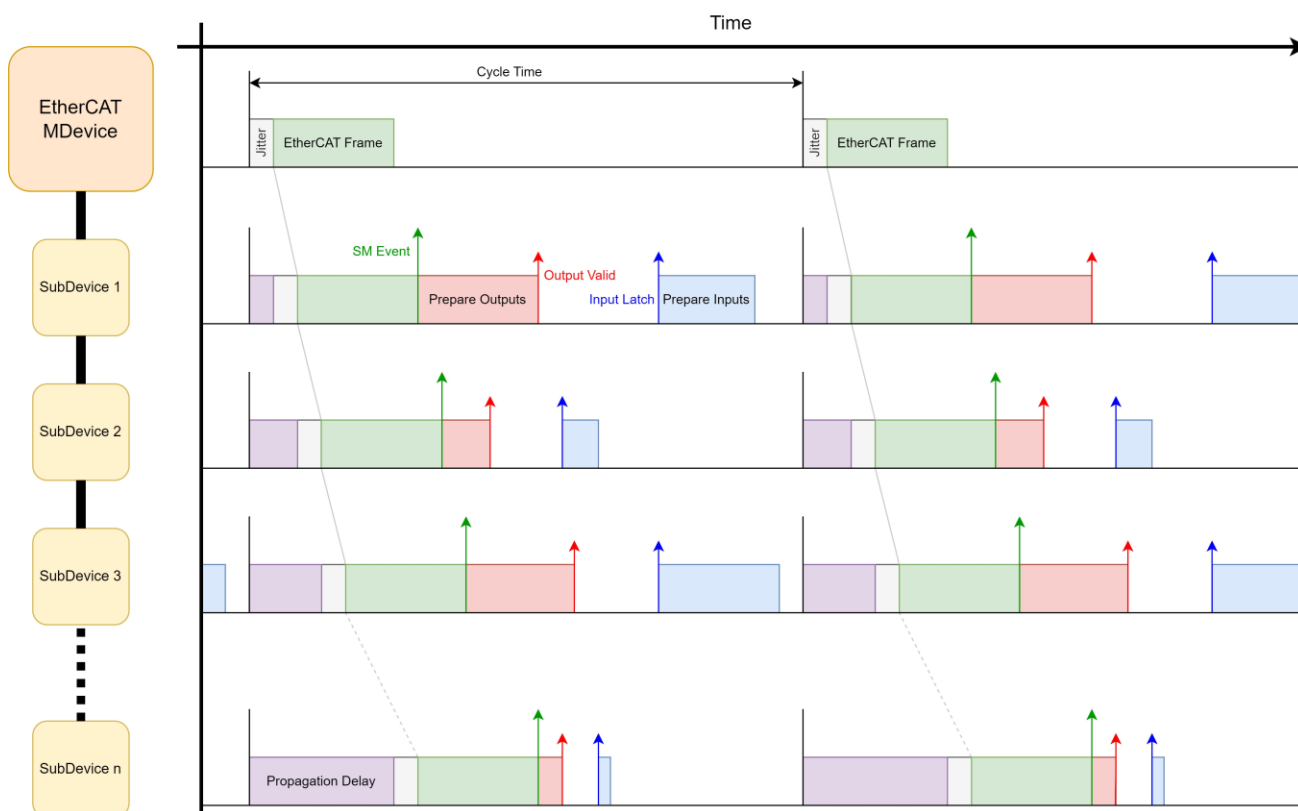


## 1.4.2 SM-Synchronous

The IRQ pin generates a SM event when the cyclic frame is received by ESC, this event will trigger the execution of the local application in  $\mu\text{C}$ . As shown in the diagram below, cyclic frames are received by SubDevices with the same jitter of MDevice in sending them. Even assuming zero jitter, due to finite hardware **Propagation Delay** the last SubDevices will receive the cyclic frames later with respect to the first ones.

Due to the Propagation Delay, there is an offset in the timing of SM events between SubDevices, resulting in an accuracy of SM-Synchronous at the **microsecond** level.

If each SubDevice supports the **Shift Time** in the **SyncManager Parameter objects** (0x1C32.3/0x1C33.3), it is possible to attempt to adjust the Output Valid and Input Latch of all SubDevices to be close to each other. However, due to the inability to calculate the propagation delays, the adjustment is quite challenging.



### 1.4.3 DC-Synchronous

The SYNC0 or SYNC1 pins generate SYNC events cyclically based on the configuration in the DC related registers of ESC, this event will trigger the execution of the local application in  $\mu$ C. As shown in the diagram below, jitters and propagation delays still exist, and SM events are still triggered after receiving cyclic frames. However, in this DC-Synchronous method, SYNC0 events are triggered cyclically, which does not suffer from jitter or propagation delays.

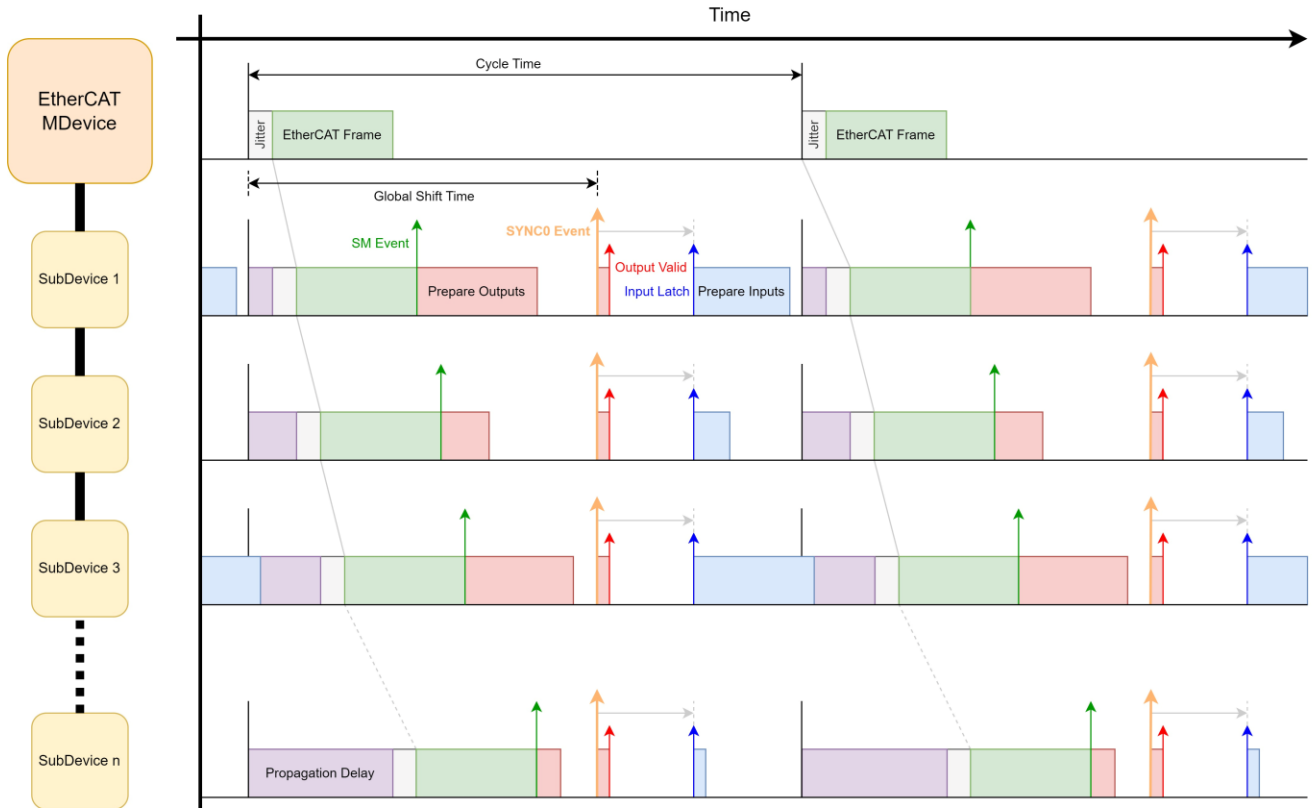
Because SYNC0 events are triggered by the DC unit and all SYNC0 events among the SubDevices have almost no offset, thanks to the periodic sending of APRW/FPRW commands to synchronize the system time of all SubDevices, the accuracy can reach the **nanosecond** level.

If the SubDevices support the Shift Time in SyncManager Parameter objects (0x1C32.3/0x1C33.3), it is possible to attempt to adjust the Output Valid and Input Latch timings of these SubDevices to the same time point.

However, the selection of the **Global Shift Time** in the diagram is crucial but must meet the following conditions:

- After the cyclic frames have been received by all SubDevices.
- Before sending the cyclic frames for the next cycle.
- According to different DC-Synchronous methods, it may need to be selected after executing Prepare Outputs:
  - Trigger  $\mu$ C to execute Prepare Outputs when the SM event occurs
  - Trigger  $\mu$ C to execute Output Valid when the SYNC event occurs.

The correct Global Shift Time is not unique; it can be chosen within the entire interval of the cycle time. To learn more about various DC-Synchronous methods, please refer to **ETG.1020 EtherCAT Protocol Enhancements**.



## 1.5 Benchmark

EtherCAT is a fieldbus technology known for its high synchronization capabilities. In applications requiring high synchronization, there is often a demand for real-time performance and high control frequencies. Users in these scenarios typically consider specifications such as:

- Support for shorter cycle times
- Support for more process data
- Support for more EtherCAT SubDevices

However, assessing whether an EtherCAT MDevice meets the user's application requirements often involves benchmark measurements as a primary consideration.

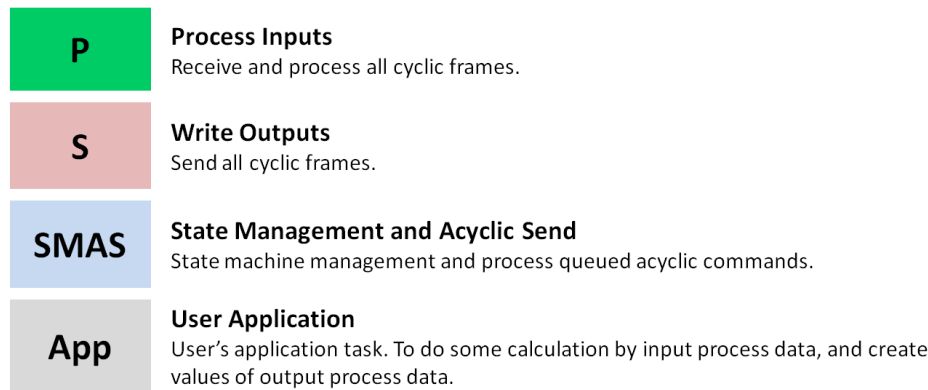
### 1.5.1 System Variables

The following factors and variables determine the achievable performance and the choice of EtherCAT cycle time:

- **Network Cable Length**  
Affects the transmission delay of network packets.
- **Number of SubDevices**  
Influences the transmission delay of network packets; each SubDevice may contribute approximately 0.3 to 1  $\mu\text{s}$  of transmission delay.
- **SubDevice Process Data Bytes**  
Determines the length of network packets.
- **SubDevice Synchronous Mode**  
Constrained by the EtherCAT SubDevice's processing performance in the application.
- **MDevice Computational Efficiency**  
The efficiency of the EtherCAT MDevice not only affects the cycle time but also influences synchronization accuracy. Factors determining the efficiency of the EtherCAT MDevice include CPU processing speed, software architecture and efficiency, memory transfer speed, etc.

## 1.5.2 Measurement Functions

For EtherCAT applications with real-time requirements, precise task scheduling within the cycle time is crucial to minimize cycle time jitter. In the QEC-MDevice software, the cycle time is divided into four phases, each with its respective tasks, as follows:

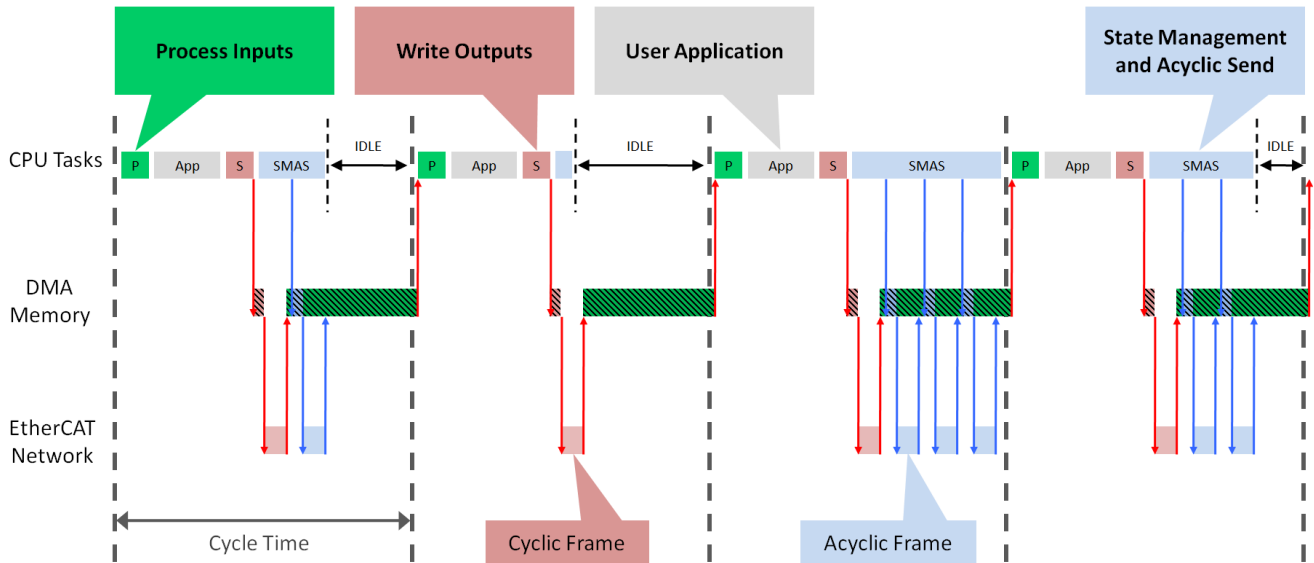


In the default cycle mode, the timing diagram for tasks within the cycle time is illustrated as follows:

- **Process Inputs**  
The first task at the beginning of the cycle is performed by the EtherCAT MDevice firmware. It is responsible for receiving and processing cyclic process data, transferring Input Process Data to the shared memory of the dual system.
- **User Application**  
The EtherCAT MDevice firmware sends a Cyclic Interrupt to the main system. The main system transfers Input Process Data from shared memory to main memory for user reading of the previous cycle's process data. Subsequently, the user's registered Cyclic Callback is invoked, allowing the user to operate EtherCAT SubDevices at the correct time in control system. The user writes the Output Process Data of current cycle to the main memory. The main system moves Output Process Data to shared memory after completion and responds to the EtherCAT MDevice firmware that the User Application has finished execution.
- **Write Outputs**  
The EtherCAT MDevice firmware performs the Write Outputs task after waiting for the main system's response or timeout. This task involves packaging Output Process Data from shared memory into EtherCAT cyclic process data frames and sending them to all SubDevices.

- **State Management and Acyclic Send**

The final task within the cycle time for the EtherCAT MDevice firmware. This task involves sending, receiving, and processing the EtherCAT state machine, as well as handling acyclic data transmission (e.g., Mailbox, CoE).



### 1.5.3 Measurement Result

The following measurement results were performed with 5 SubDevices on the same controllers with different cycle times.

The 5 SubDevices are as follows:

- QEC-R00DC4D: 12 digital inputs, and 4 digital outputs.
- QEC-R00D4CD: 4 digital inputs, and 12 digital outputs.
- QEC-R00D88D: 8 digital inputs, and 8 digital outputs.
- QEC-R00D0FS: 16 digital outputs.
- QEC-R00DF0D: 16 digital inputs.

The platform conditions are as follows:

- Processor: Vortex86EX2 600/400 MHz
- OS: FreeDOS
- Compiler: GCC 8.3.0
- Payload: 60 Bytes

#### 1.5.3.1 Configured Cycle Time equals 125 $\mu$ s

##### With Acyclic Transfer

MDevice Function	min.	avg.	max.
Measured cycle time.	124.90	124.92	125.09
Process Inputs.	6.36	7.26	10.81
Write Outputs.	7.22	7.35	11.89
State Management and Acyclic Send.	2.46	2.48	34.89

##### Without Acyclic Transfer

The maximum throughput of SDO commands per second is **769.48**.

MDevice Function	min.	avg.	max.
Measured cycle time.	124.85	124.86	125.12
Process Inputs.	4.79	4.88	9.80
Write Outputs.	7.91	8.08	14.06
State Management and Acyclic Send.	2.47	60.55	96.30

### 1.5.3.2 Configured Cycle Time equals 250 $\mu$ s

#### With Acyclic Transfer

MDevice Function	min.	avg.	max.
Measured cycle time.	249.92	249.93	250.11
Process Inputs.	5.99	7.83	10.42
Write Outputs.	7.90	7.92	12.29
State Management and Acyclic Send.	2.33	2.33	26.66

#### Without Acyclic Transfer

The maximum throughput of SDO commands per second is **1409.11**.

MDevice Function	min.	avg.	max.
Measured cycle time.	249.86	249.87	250.19
Process Inputs.	5.42	5.50	13.03
Write Outputs.	8.39	8.66	13.27
State Management and Acyclic Send.	2.90	12.27	227.15

### 1.5.3.3 Configured Cycle Time equals 500 $\mu$ s

#### With Acyclic Transfer

MDevice Function	min.	avg.	max.
Measured cycle time.	499.93	499.93	500.08
Process Inputs.	6.13	7.91	10.26
Write Outputs.	7.91	7.98	12.50
State Management and Acyclic Send.	2.28	2.34	35.17

#### Without Acyclic Transfer

The maximum throughput of SDO commands per second is **1657.79**.

MDevice Function	min.	avg.	max.
Measured cycle time.	499.81	499.83	500.19
Process Inputs.	5.71	6.01	14.03
Write Outputs.	8.84	9.04	14.14
State Management and Acyclic Send.	2.77	123.89	474.36

### 1.5.3.4 Configured Cycle Time equals 1000 $\mu$ s

#### With Acyclic Transfer

MDevice Function	min.	avg.	max.
Measured cycle time.	999.95	999.96	1000.07
Process Inputs.	6.57	8.29	10.11
Write Outputs.	7.99	8.01	11.72
State Management and Acyclic Send.	2.32	2.33	34.71

#### Without Acyclic Transfer

The maximum throughput of SDO commands per second is **999.99**.

MDevice Function	min.	avg.	max.
Measured cycle time.	999.93	999.94	1000.08
Process Inputs.	6.42	6.60	9.18
Write Outputs.	10.32	11.57	14.03
State Management and Acyclic Send.	9.15	345.94	593.68

## 1.5.4 Example Code

Here is an example code for benchmark measurement, testing with cycle time of 1000 microseconds.

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  EthercatMasterSettings settings;
  EthercatBenchmark benchmark;

  Serial.begin(115200);

  master.readSettings(&settings);
  settings.EnableBenchmarkMeasurement = 1;
  master.saveSettings(&settings);

  if (master.begin() < 0) {
    Serial.println("ERROR: master.begin() failed.");
    return;
  }

  if (master.start(1000000) < 0) {
    Serial.println("ERROR: master.start() failed.");
    master.end();
    return;
  }

  delay(30000);

  master.getBenchmarkResult(&benchmark);
  Serial.println();

  Serial.println(" |=====
=====|");
  Serial.print(" | [C]   Cycle Time.      (min/avg/max) [usec]: ");
  Serial.print(benchmark.CycleTime.min / 1000.0, 2); Serial.print(" / ");
  Serial.print(benchmark.CycleTime.avg / 1000.0, 2); Serial.print(" / ");
  Serial.print(benchmark.CycleTime.max / 1000.0, 2); Serial.println(" |");
```

```

Serial.print("| [P]   Receive PDO.      (min/avg/max) [usec]: ");
Serial.print(benchmark.ReceiveCyclicFrame.min / 1000.0, 2); Serial.print("
/ ");
Serial.print(benchmark.ReceiveCyclicFrame.avg / 1000.0, 2); Serial.print("
/ ");
Serial.print(benchmark.ReceiveCyclicFrame.max / 1000.0, 2);
Serial.println(" |");

Serial.print("| [S]   Send PDO.          (min/avg/max) [usec]: ");
Serial.print(benchmark.SendCyclicFrame.min / 1000.0, 2); Serial.print(" /
");
Serial.print(benchmark.SendCyclicFrame.avg / 1000.0, 2); Serial.print(" /
");
Serial.print(benchmark.SendCyclicFrame.max / 1000.0, 2); Serial.println("
|");

Serial.print("| [AS]  Process Acyclic.  (min/avg/max) [usec]: ");
Serial.print(benchmark.ProcessAcyclicFrame.min / 1000.0, 2); Serial.print("
/ ");
Serial.print(benchmark.ProcessAcyclicFrame.avg / 1000.0, 2); Serial.print("
/ ");
Serial.print(benchmark.ProcessAcyclicFrame.max / 1000.0, 2);
Serial.println(" |");

Serial.print("| [App.] User Application. (min/avg/max) [usec]: ");
Serial.print(benchmark.UserApp.min / 1000.0, 2); Serial.print(" / ");
Serial.print(benchmark.UserApp.avg / 1000.0, 2); Serial.print(" / ");
Serial.print(benchmark.UserApp.max / 1000.0, 2); Serial.println(" |");

Serial.println(" |=====  

=====  

|");

    master.stop();
    master.end();
}

void loop() {
}

```

## 1.5.5 EtherCAT MDevice Cyclic Frame Jitter

The jitter in the transmission of cyclic frame from the EtherCAT MDevice, which is referenced to DC SYNC0.

- Video: <https://youtu.be/O888jD4XUsY?si=Nal9gsafyA1D2DIK>

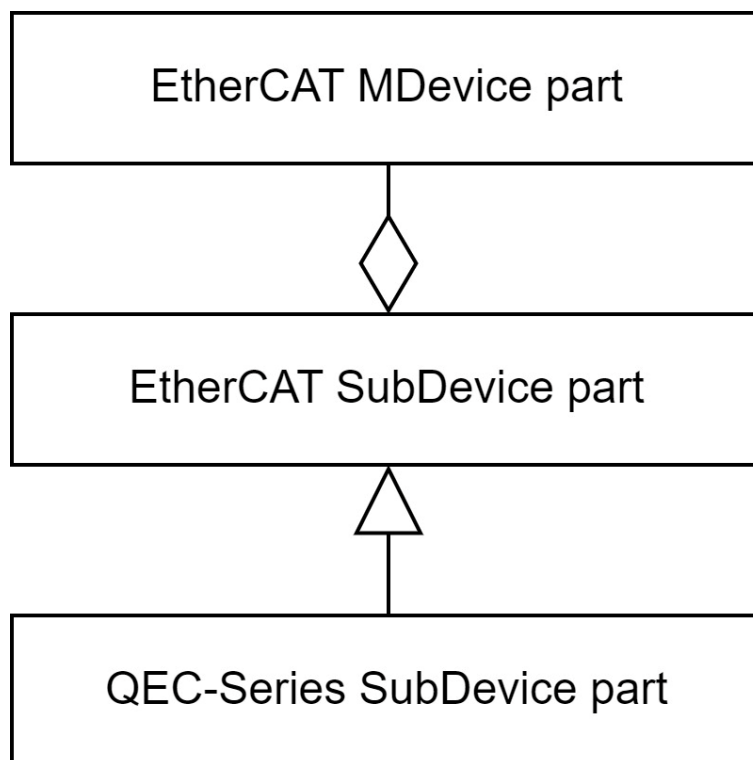




# Ch. **2**

## Functions

EtherCAT is a real-time industrial Ethernet communication protocol widely used in automation and control systems. QEC-MDevice is an EtherCAT MDevice library implemented in C/C++, which includes classes for the MDevice, generic SubDevice, CiA 402 SubDevice, and dedicated classes for QEC series SubDevices. These classes not only have clearly defined responsibilities but also consider future extensibility.



These classes can be divided into three parts as follows:

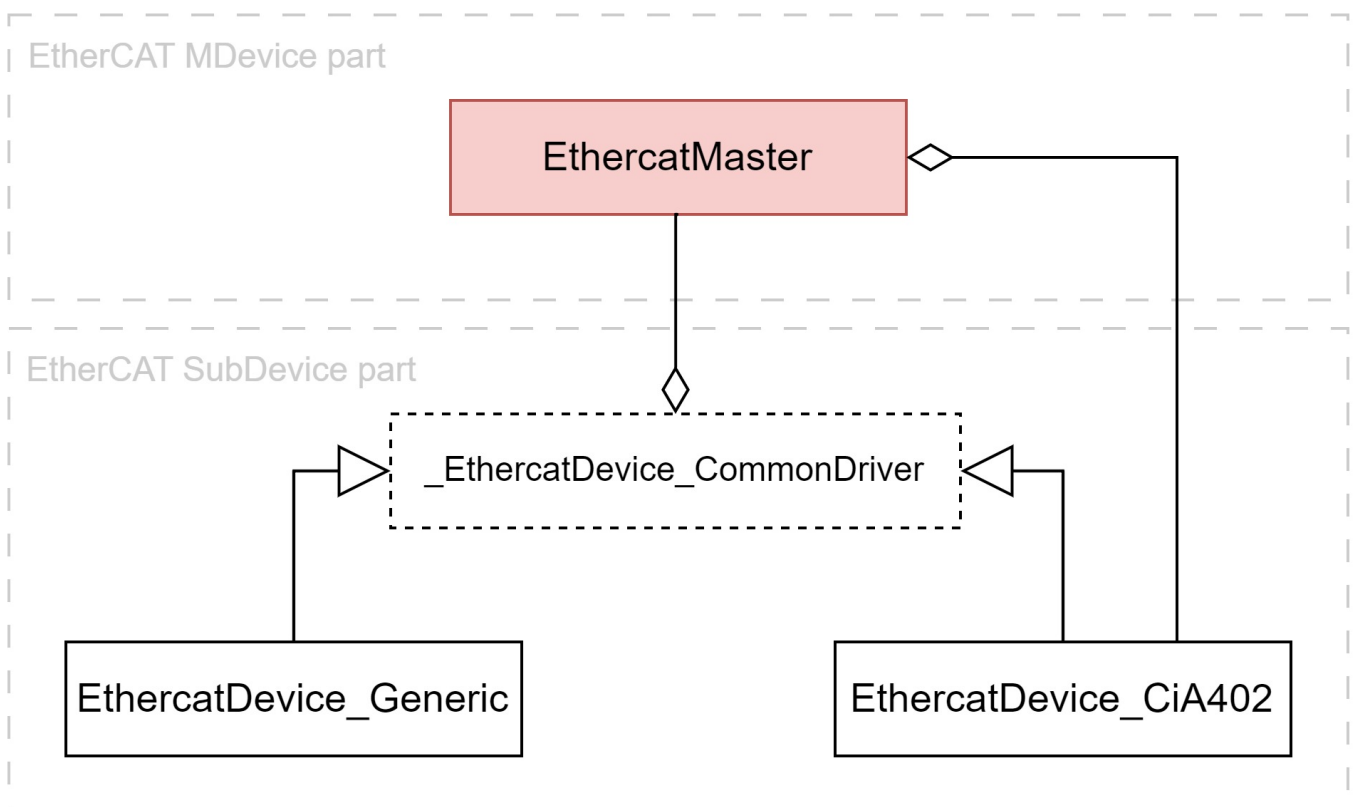
- [EtherCAT MDevice](#)  
The EtherCAT MDevice part not only provides various and flexible MDevice configuration and operation functions but also offers diverse EtherCAT SubDevice operation functions for invocation by the EtherCAT SubDevice part.
- [EtherCAT SubDevice](#)  
The EtherCAT SubDevice part provides generic EtherCAT SubDevice classes, which can operate functions such as PDOs, CoE, FoE, and also includes CiA 402 SubDevice generic class.
- [QEC-Series SubDevice](#)  
The QEC-Series SubDevice part provides dedicated functions for ICOP's QEC series SubDevices, enabling users to code in a more user-friendly and concise manner.

## 2.1 EtherCAT MDevice

The EtherCAT MDevice part not only provides various and flexible MDevice configuration and operation functions but also offers diverse EtherCAT SubDevice operation functions for invocation by the EtherCAT SubDevice part.

**EthercatMaster** is the only class in the EtherCAT MDevice part, it serves as a crucial communication bridge with the EtherCAT firmware. In the Dual-System communication aspect, its responsibilities include communication interface initialization, process data exchange cyclically, handling acyclic transfer interfaces, and managing interrupt events. In the API aspect, it provides functions related to MDevice initialization, MDevice control, and access to SubDevice information.

The main class relationship between the EtherCAT MDevice part and the EtherCAT SubDevice part is association, with the EtherCAT SubDevice part depending on the EtherCAT MDevice part. The class relationships of EthercatMaster are illustrated in the following diagram:



- There is an association between `EthercatMaster` and `_EthercatDevice_CommonDriver`, with `_EthercatDevice_CommonDriver` depending on `EthercatMaster`.
- There is an association between `EthercatMaster` and `EthercatDevice_CiA402`, with `EthercatMaster` depending on `EthercatDevice_CiA402`.

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">begin()</a>	Initialize the EtherCAT MDevice.	
<a href="#">end()</a>	Deinitialize the EtherCAT MDevice.	
<a href="#">isRedundancy()</a>	Check if the EtherCAT MDevice has cable redundancy enabled.	0
<a href="#">libraryVersion()</a>	Get the EtherCAT MDevice library version.	0
<a href="#">firmwareVersion()</a>	Get the EtherCAT firmware version.	0
<a href="#">readSettings()</a>	Read the current EtherCAT MDevice settings.	
<a href="#">saveSettings()</a>	Save the EtherCAT MDevice settings.	
<b>Control-related functions</b>		
<a href="#">start()</a>	Start the EtherCAT MDevice.	
<a href="#">stop()</a>	Stop the EtherCAT MDevice.	
<a href="#">update()</a>	Update process data and handle acyclic commands.	0
<a href="#">setShiftTime()</a>	Set the Global Shift Time for DC-Synchronous mode.	
<a href="#">getShiftTime()</a>	Get the Global Shift Time for DC-Synchronous mode.	0
<a href="#">getSystemTime()</a>	Get the system time of the current cycle.	0
<a href="#">getWorkingCounter()</a>	Get the working counter for the current cycle.	0
<a href="#">getExpectedWorkingCounter()</a>	Get the expected working counter.	0
<b>Callback-related functions</b>		
<a href="#">attachCyclicCallback()</a>	Register a cyclic callback.	
<a href="#">detachCyclicCallback()</a>	Unregister cyclic callback.	
<a href="#">attachErrorCallback()</a>	Register an error callback.	
<a href="#">detachErrorCallback()</a>	Unregister error callback.	
<a href="#">attachEventCallback()</a>	Register an event callback.	
<a href="#">detachEventCallback()</a>	Unregister event callback.	
<a href="#">errGetCableBrokenLocation1()</a>	Get the cable broken location 1 in error callback.	0 <sup>1</sup>
<a href="#">errGetCableBrokenLocation2()</a>	Get the cable broken location 2 in error callback.	0 <sup>1</sup>
<a href="#">evtGetMasterState()</a>	Get the EtherCAT MDevice state in event callback.	0 <sup>2</sup>
<b>SubDevice information related functions</b>		
<a href="#">getSlaveCount()</a>	Get the number of SubDevices on the network.	0
<a href="#">getVendorID()</a>	Get the vendor ID of the specified device.	0
<a href="#">getProductCode()</a>	Get the product code of the specified device.	0
<a href="#">getRevisionNumber()</a>	Get the revision number of the specified device.	0
<a href="#">getSerialNumber()</a>	Get the serial number of the specified device.	0
<a href="#">getAliasAddress()</a>	Get the alias address of the specified device.	0
<a href="#">getSlaveNo()</a>	Find the sequence number of the matching EtherCAT SubDevice on the network.	0

- **Note 1:** This function can only be called in error callback.
- **Note 2:** This function can only be called in event callback.

## Function Groups:

- [Initialization](#)
- [Control](#)
- [Callback](#)
- [SubDevice Information](#)

## 2.1.1 EtherCAT MDevice Settings

This library offers a variety of configuration parameters for users to choose from, aiming to meet the diverse application needs of users. Below are the configuration parameters provided by this library.

```
typedef struct {  
  
    EthercatDcSyncMode DcSyncMode;  
    uint32_t StaticDriftCompensationFrames;  
  
    uint32_t StateMachineTimeoutI2P;  
    uint32_t StateMachineTimeoutP2S;  
    uint32_t StateMachineTimeoutS2O;  
    uint32_t ScanNetworkTimeout;  
    uint32_t StartMasterTimeout;  
    uint32_t StartDeviceTimeout;  
  
    uint32_t ErrorDetectWkcMultipleFaultsThreshold;  
    uint32_t ErrorDetectMultipleLostFramesThreshold;  
    uint32_t EnableErrorBusReactionSyncUnitToSafeOp:1,  
            EnableErrorBusReactionSyncUnitToSafeOpAutoRestart:1,  
            IgnoreBiosOverride:1;  
  
} EthercatMasterSettings;
```

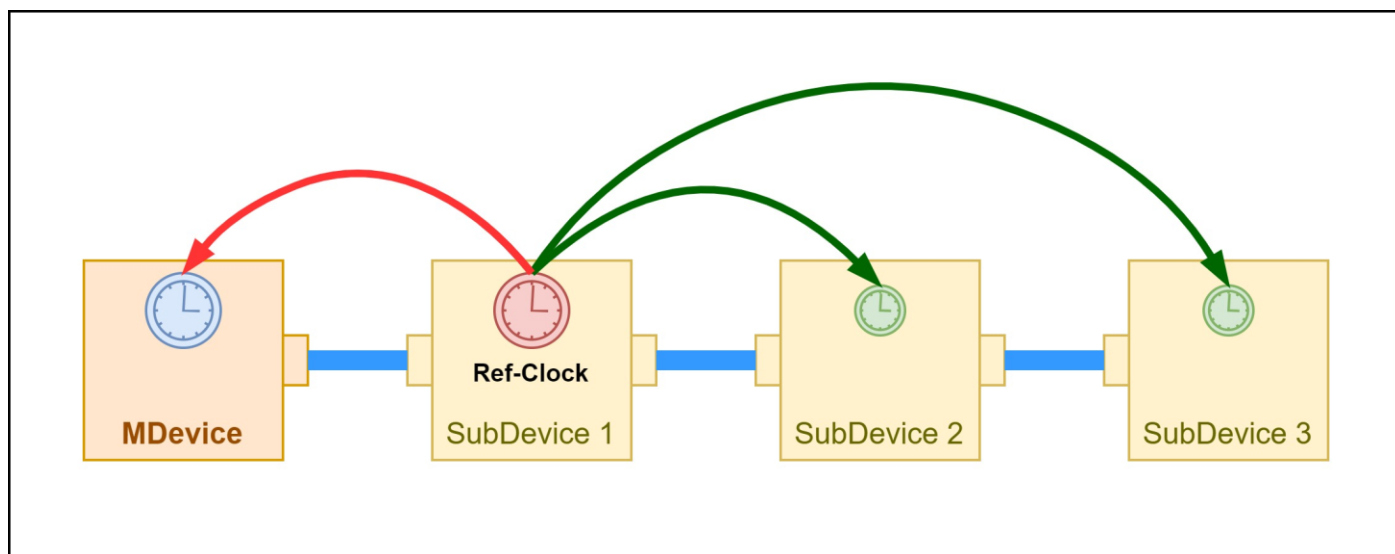
## DcSyncMode

Default: `ECAT_MASTER_SHIFT`.

In DC-Synchronous mode, the first SubDevice with DC serves as the system reference clock to synchronize other SubDevices with DC. However, this only involves synchronizing the system time of all SubDevices on the network and does not include the EtherCAT MDevice. In applications with DC-Synchronous mode enabled, the MDevice usually needs to precisely and periodically control the SubDevices, so the MDevice must also synchronize its system time with all SubDevices on the network.

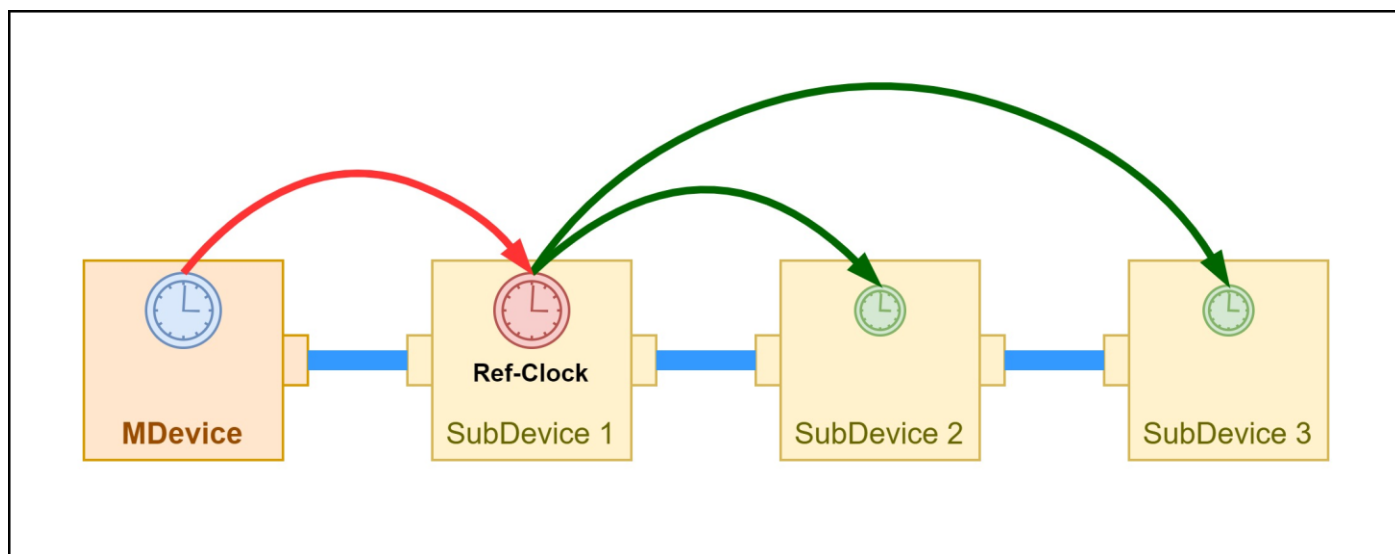
There are two methods for this synchronization:

- **MDevice Shift Mode** - `ECAT_MASTER_SHIFT`



- The MDevice system time is synchronized to the reference clock.
- All DC SubDevices are synchronized to the reference clock.

- **Bus Shift Mode** - `ECAT_BUS_SHIFT`



- The reference clock is synchronized to the MDevice system time.
- All DC SubDevices are synchronized to the reference clock.

## StaticDriftCompensationFrames

Default: 30000; Unit: frames.

The EtherCAT MDevice sends many separate ARMW or FRMW drift compensation frames to distribute the System Time of the reference clock to all DC SubDevices.

## StateMachineTimeoutI2P

Default: 3000; Unit: milliseconds.

Timeout for the transition from the Init state to the Pre-Operational state.

## StateMachineTimeoutP2S

Default: 10000; Unit: milliseconds.

Timeout for the transition from the Pre-Operational state to the Safe-Operational state.

## StateMachineTimeoutS2O

Default: 10000; Unit: milliseconds.

Timeout for the transition from the Safe-Operational state to the Operational state.

## ScanNetworkTimeout

Default: 5000; Unit: milliseconds.

Timeout for scan network.

The scan network operation is executed within [EthercatMaster::begin\(\)](#).

## StartMasterTimeout

Default: 3000; Unit: milliseconds.

Base timeout for start MDevice,  $T_{base}$ .

In [EthercatMaster::start\(\)](#), the firmware is requested to start EtherCAT, and the timeout for this request is referred to as startup timeout,  $T_{startup}$ .

The startup timeout for EtherCAT is calculated as follows:

$$T_{startup} = T_{base} + (T_{SubDevice} \times N_{SubDevices})$$

Here,  $N_{SubDevices}$  is the number of SubDevices on the network.

## StartDeviceTimeout

Default: 500; Unit: milliseconds.

Timeout per SubDevice for start MDevice,  $T_{SubDevice}$ .

## ErrorDetectWkcMultipleFaultsThreshold

Default: 3.

The MDevice should check the Working Counter of a received EtherCAT datagram. If the Working Counter does not match with the expected value an error is detected. When the number of consecutive errors exceeds this parameter, an [ECAT\\_ERR\\_WKC\\_MULTIPLE\\_FAULTS](#) error interrupt will be triggered.

## ErrorDetectMultipleLostFramesThreshold

Default: 3.

The MDevice may use the index of the EtherCAT datagram header to check if all sent EtherCAT datagrams will be received. If EtherCAT datagrams are lost an error is detected. When the number of consecutive errors exceeds this parameter, an [ECAT\\_ERR\\_MULTIPLE\\_LOST\\_FRAMES](#) error interrupt will be triggered.

## EnableErrorBusReactionSyncUnitToSafeOp

Default: 0.

If this parameter is set to 1, the MDevice will change the EtherCAT state of the SubDevices with an application controller and will disable the Sync Manager channels of the SubDevices which only support the EtherCAT state machine emulation.

## EnableErrorBusReactionSyncUnitToSafeOpAutoRestart

Default: 1.

This parameter only takes effect if `EnableErrorBusReactionSyncUnitToSafeOp` is set to 1. If this parameter is set to 1, the MDevice will automatically attempt to restart the Sync Unit according to the Restart Behavior of a Sync Unit in the MDevice as defined in ETG.1020 EtherCAT Protocol Enhancements, switching the EtherCAT state machine back to the Operational state.

## IgnoreBiosOverride

Default: 0.

QEC-MDevice has some EtherCAT configuration parameters in the BIOS. Setting this parameter to 1 means ignoring the EtherCAT configuration parameters in the BIOS; otherwise, they are not ignored.

## 2.1.2 Initialization Functions

Before starting the EtherCAT MDevice, it must be initialized.

This library offers a variety of configuration parameters for users to choose from, aiming to meet the diverse application needs of users.

Functions:

- [begin\(\)](#)
- [end\(\)](#)
- [isRedundancy\(\)](#)
- [libraryVersion\(\)](#)
- [firmwareVersion\(\)](#)
- [readSettings\(\)](#)
- [saveSettings\(\)](#)

begin()

### Description

Initialize the EtherCAT MDevice, scan all EtherCAT SubDevices on the network, and switch the EtherCAT state machine to the Pre-Operational state.

### Syntax

```
int begin(EthernetPort eth = ECAT_ETH_0, const char *eni_filename = NULL);
```

### Parameters

- **[in] eth**  
Selection of the Ethernet interface for EtherCAT communication.  
**ECAT\_ETH\_0**: Only eth0 is used as the EtherCAT communication interface.  
**ECAT\_ETH\_1**: Only eth1 is used as the EtherCAT communication interface.  
**ECAT\_ETH\_REDUNDANCY**: Enable EtherCAT Cable Redundancy with eth0 as the primary port and eth1 as the secondary port.  
The default is **ECAT\_ETH\_0**.
- **[in] eni\_filename**  
The file name of the EtherCAT Network Information (ENI). For details about the ENI content supported by this library, please refer to [EtherCAT Network Information](#).  
The default is NULL.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  master.begin();
  // ...
  master.start(1000000); // cycle time set as 1 millisecond.
}

void loop() {
}
```

end()

**Description**

Deinitialize the EtherCAT MDevice.

**Syntax**

```
void end();
```

**Parameters**

None.

**Return Value**

None.

**Comment**

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

**Example Code**

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    master.begin();
    // ...
    master.end();
}

void loop() {

}
```

## isRedundancy()

### Description

Check if the EtherCAT MDevice has cable redundancy enabled.

### Syntax

```
bool isRedundancy();
```

### Parameters

None.

### Return Value

Return whether Cable Redundancy is enabled for the EtherCAT MDevice.

`true` indicates it is enabled, while `false` indicates it is not.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  Serial.println(master.isRedundancy());
  // ...
}

void loop() {
  // ...
}
```

## libraryVersion()

### Description

Get the EtherCAT MDevice library version.

### Syntax

```
char *libraryVersion();
```

### Parameters

None.

### Return Value

Return a pointer to the EtherCAT MDevice library version string.

### Comment

This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  Serial.print("EtherCAT Library Version: ");
  Serial.println(master.libraryVersion());
  // ...
}

void loop() {
  // ...
}
```

## firmwareVersion()

### Description

Get the EtherCAT firmware version.

### Syntax

```
char *firmwareVersion();
```

### Parameters

None.

### Return Value

Return a pointer to the EtherCAT firmware version string.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();

  Serial.print("EtherCAT Firmware Version: ");
  Serial.println(master.firmwareVersion());
  // ...
}

void loop() {
}
```

## readSettings()

### Description

Read the current EtherCAT MDevice settings.

### Syntax

```
int readSettings(EthercatMasterSettings *settings);
```

### Parameters

- `[in] settings`  
A pointer to the `EthercatMasterSettings` data structure. For more details about the `EthercatMasterSettings` parameters, please refer to [EtherCAT MDevice Settings](#).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  EthercatMasterSettings settings;

  master.readSettings(&settings);
  settings.DcSyncMode = ECAT_BUS_SHIFT;
  master.saveSettings(&settings);

  master.begin();

  // ...
}

void loop() {
}
```

## saveSettings()

### Description

Save the EtherCAT MDevice settings.

### Syntax

```
int saveSettings(EthercatMasterSettings *settings);
```

### Parameters

- `[in] settings`  
A pointer to the `EthercatMasterSettings` data structure. For more details about the `EthercatMasterSettings` parameters, please refer to [EtherCAT MDevice Settings](#).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  EthercatMasterSettings settings;

  master.readSettings(&settings);
  settings.DcSyncMode = ECAT_BUS_SHIFT;
  master.saveSettings(&settings);

  master.begin();

  // ...
}

void loop() {
}
```

## 2.1.3 Control Functions

The control functions provided by the EtherCAT MDevice library are crucial for managing the state and operation of the EtherCAT network.

By using these functions, users can ensure precise control over the network, achieving reliable and synchronized communication between the MDevice and SubDevices.

Functions:

- [start\(\)](#)
- [stop\(\)](#)
- [update\(\)](#)
- [setShiftTime\(\)](#)
- [getShiftTime\(\)](#)
- [getSystemTime\(\)](#)
- [getWorkingCounter\(\)](#)
- [getExpectedWorkingCounter\(\)](#)

start()

### Description

Start the EtherCAT MDevice, configure the SM, FMMU, and DC registers of all SubDevices, and switch the EtherCAT state machine to the Operational state.

### Syntax

```
int start(uint64_t cycletime_ns = 0, EthercatMasterMode mode = ECAT_FREERUN);
```

### Parameters

- **[in] cycletime\_ns**  
EtherCAT cycle time, unit is nanoseconds.  
The EtherCAT firmware will periodically generate interrupts according to this cycle time to update process data and handle acyclic transmissions. If the user has registered a Cyclic Callback and the EtherCAT MDevice control mode is not set to **ECAT\_FREERUN\_MANUAL**, the callback will be called during these periodic interrupts.  
The default is 1000000 ns (1 millisecond).
- **[in] EthercatMasterMode mode**  
Selection of the EtherCAT control mode. For detailed descriptions of each mode, please refer to the examples below.
  - a. **ECAT\_SYNC**
  - b. **ECAT\_FREERUN**
  - c. **ECAT\_FREERUN\_MANUAL**
 The default is **ECAT\_FREERUN**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

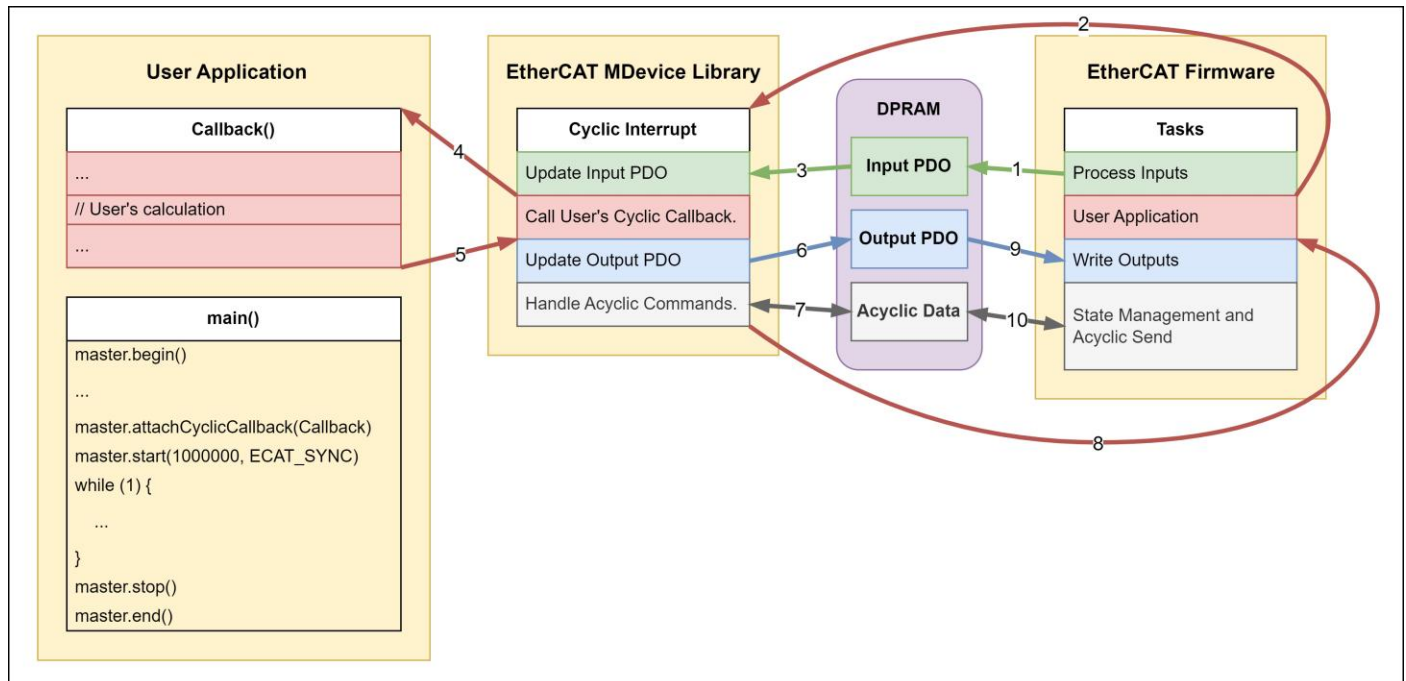
This function is blocking and cannot be called in callback functions.

### Example Code

- A. [ECAT\\_SYNC](#)
- B. [ECAT\\_FREERUN](#)
- C. [ECAT\\_FREERUN\\_MANUAL](#)

## A. ECAT\_SYNC

This mode offers the highest level of dual-system synchronization.



As shown in the diagram, the numbered arrows indicate the sequence of operations, with no branching present. After the EtherCAT firmware triggers a cyclic interrupt to the EtherCAT MDevice library (step 2), it waits for an ACK response from the EtherCAT MDevice library (step 8) before proceeding with the next action. If the user has registered a cyclic callback, the cyclic interrupt will call it, as shown in step 4. As long as the user reads the current input process data in the cyclic callback, processes it, calculates the output process data, and writes it back, the current cycle will send the output process data to the EtherCAT network, fulfilling the requirements of real-time control systems.

Here is the example code.

```

#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

void setup() {
  Serial.begin(115200);

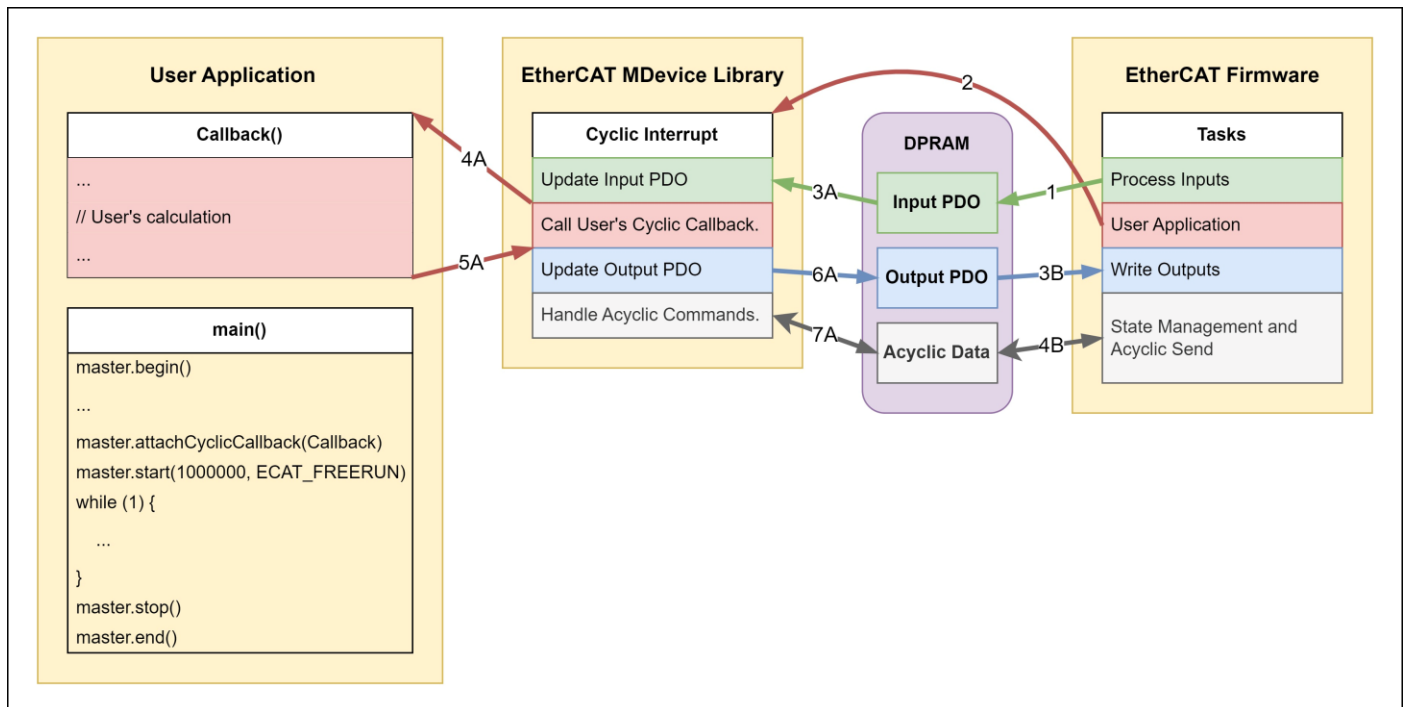
  master.begin();
  
```

```
master.attachCyclicCallback(CyclicCallback);
master.start(1000000, ECAT_SYNC);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## B. ECAT\_FREERUN

This is the free-run mode without dual-system synchronization.



As shown in the diagram, the numbered arrows indicate the sequence of operations. However, a branching occurs at step 3 because, after the EtherCAT firmware triggers a cyclic interrupt to the EtherCAT MDevice library (step 2), it does not wait for the EtherCAT MDevice library and directly continues with the next action. The two systems operate independently, with no synchronization. If the user has registered a cyclic callback, the cyclic interrupt will call it, as shown in step 4A.

Here is the example code.

```

#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

void setup() {
  Serial.begin(115200);

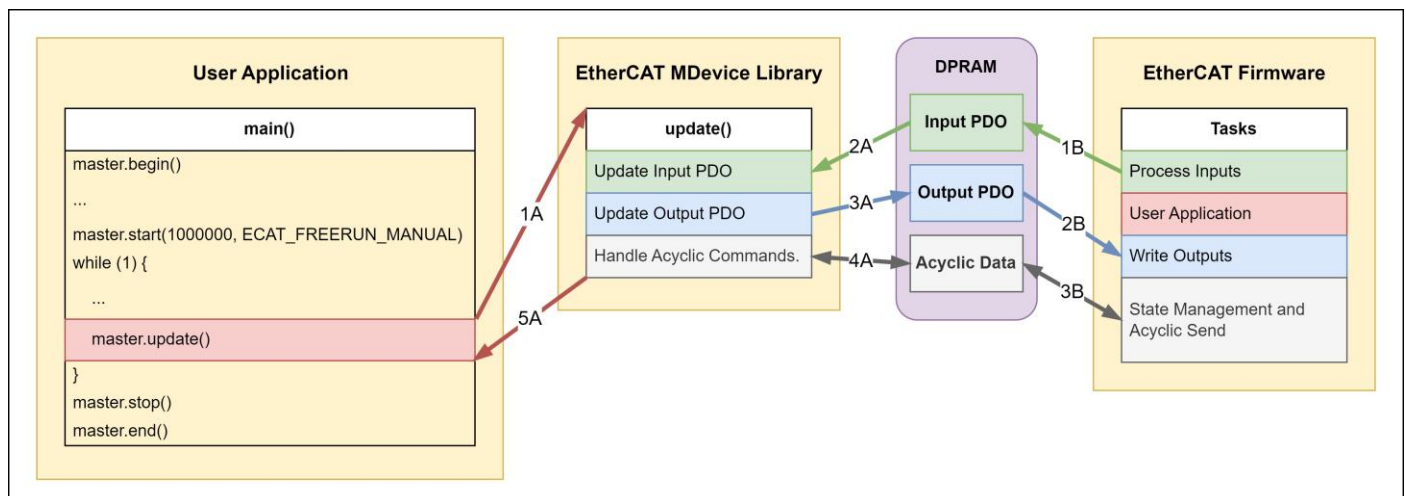
  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  master.start(1000000, ECAT_FREERUN);
}

```

```
void loop() {  
  // ...  
}
```

## C. ECAT\_FREERUN\_MANUAL

This is also a free-run mode without dual-system synchronization.



The primary difference from the *ECAT\_FREERUN* mode is that there is no cyclic interrupt to update process data and handle acyclic commands. Instead, the user must manually call `EthercatMaster::update()` to update process data and handle acyclic commands. Additionally, since there is no cyclic interrupt in this mode, the cyclic callback will not be called. As indicated by the numbered arrows in the diagram, the two systems operate independently, with no synchronization.

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  master.start(1000000, ECAT_FREERUN_MANUAL);
}

void loop() {
  // ...
  master.update();
}
```

## stop()

### Description

Stop the EtherCAT MDevice, and switch the EtherCAT state machine to the Pre-Operational state.

### Syntax

```
int stop();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
    master.begin();
    master.start();
    // ...
    master.stop();
    master.end();
}

void loop() {

}
```

## update()

### Description

Update process data and handle acyclic commands. This can only operate under the **ECAT\_FREERUN\_MANUAL** control mode.

### Syntax

```
void update();
```

### Parameters

None.

### Return Value

None.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  master.begin();
  master.start(1000000, ECAT_FREERUN_MANUAL);
}

void loop() {
  // ...
  master.update();
}
```

## setShiftTime()

### Description

Set the Global Shift Time for DC-Synchronous mode. If the user does not set the Global Shift Time or sets it to `INT32_MAX`, the EtherCAT firmware will automatically calculate an appropriate value. For the definition of Global Shift Time, please refer to [Synchronization](#).

### Syntax

```
int setShiftTime(int32_t nanoseconds);
```

### Parameters

- `[in] nanoseconds`  
The value of the Global Shift Time to be set, in nanoseconds.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#) and before [EthercatMaster::start\(\)](#), or after a successful execution of [EthercatMaster::stop\(\)](#) and before [EthercatMaster::end\(\)](#). This function is non-blocking and can be in callback functions, but it is not recommended.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  device.setDc(1000000);
  master.setShiftTime(250000);
  master.start(1000000);
}

void loop() {
}
```

## getShiftTime()

### Description

Get the Global Shift Time for DC-Synchronous mode. For the definition of Global Shift Time, please refer to [Synchronization](#).

### Syntax

```
int getShiftTime();
```

### Parameters

None.

### Returns

Return the Global Shift Time, in nanoseconds.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  device.setDc(1000000);
  master.start(1000000);

  Serial.print("Global Shift Time: ");
  Serial.println(master.getShiftTime());
}

void loop() {
}
```

## getSystemTime()

### Description

Get the system time of the current cycle. It is recommended to use in the cyclic callback to ensure the system time is retrieved in the correct cycle.

### Syntax

```
uint64_t getSystemTime();
```

### Parameters

None.

### Return Value

Return the system time for the current cycle.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
uint64_t CurrentSystemTime = 0;

void CyclicCallback() {
    CurrentSystemTime = master.getSystemTime();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    char buf[32];
    snprintf(buf, sizeof(buf), "%llu", (unsigned long long)CurrentSystemTime);

    Serial.print("Current System Time: ");
    Serial.println(buf);
}
```

```
delay(1000);  
}
```

## getWorkingCounter()

### Description

Get the working counter for the current cycle.

It is recommended to use in the cyclic callback to ensure the working counter is retrieved in the correct cycle.

### Syntax

```
int getWorkingCounter();
```

### Parameters

None.

### Return Value

Return the working counter for the current cycle.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
int CurrentWorkingCounter = 0;

void CyclicCallback() {
    CurrentWorkingCounter = master.getWorkingCounter();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("Current Working Counter: ");
    Serial.println(CurrentWorkingCounter);
    delay(1000);
}
```

## getExpectedWorkingCounter()

### Description

Get the expected working counter.

### Syntax

```
int getExpectedWorkingCounter();
```

### Parameters

None.

### Return Value

Return the expected working counter.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

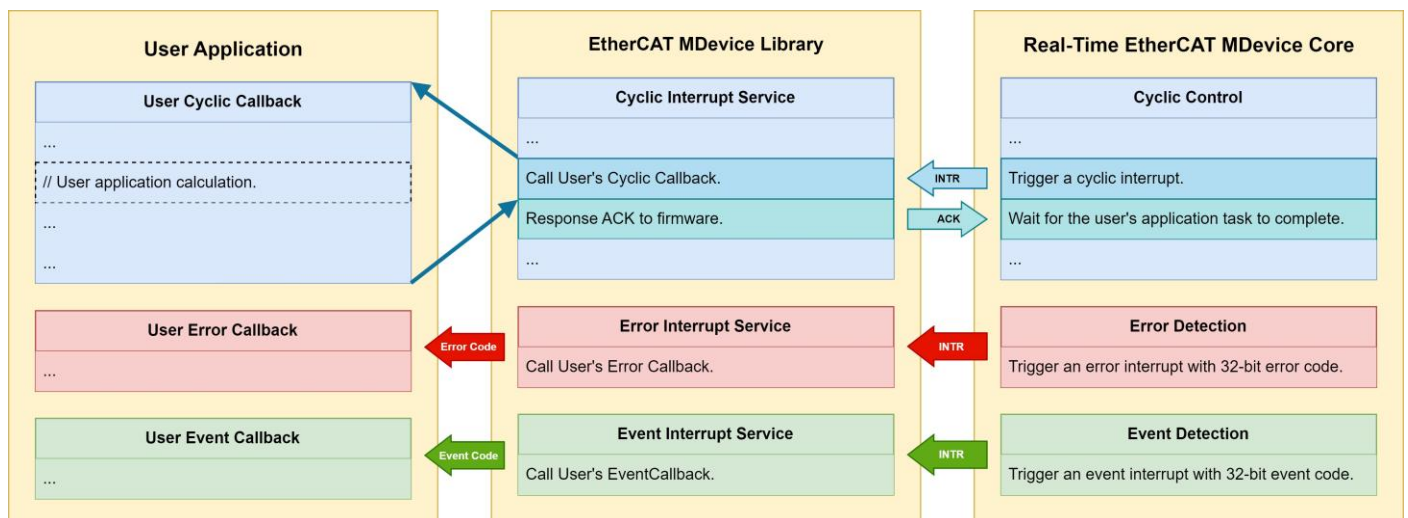
void setup() {
  Serial.begin(115200);

  master.begin();
  master.start();

  Serial.print("Expected Working Counter: ");
  Serial.println(master.getExpectedWorkingCounter());
  // ...
}

void loop() {
}
```

## 2.1.4 Callback Functions



This library provides three types of callbacks as follows:

- **Cyclic Callback**

The purpose of the Cyclic Callback is to allow users to implement periodic control systems such as motion control, CNC control, and robot control. The Real-Time EtherCAT MDevice Core triggers cyclic interrupts to the EtherCAT MDevice Library at specified cycle time, then waiting for an ACK to ensure process data synchronization. If a user has registered a Cyclic Callback, it will be invoked to achieve periodic control.

- **Error Callback**

When the Real-Time EtherCAT MDevice Core detects an error, it will trigger an error interrupt and pass a 32-bit error code to the EtherCAT MDevice Library. If the user has registered an error callback, the system will invoke that callback to inform the user of the specific error. The error codes supported by the Error Callback are as follows:

Definition	Code	Description
ECAT_ERR_WKC_SINGLE_FAULT	2000001	Working counter fault occurred.
ECAT_ERR_WKC_MULTIPLE_FAULTS	2000002	Multiple working counter faults occurred.
ECAT_ERR_SINGLE_LOST_FRAME	2000003	Frame was lost.
ECAT_ERR_MULTIPLE_LOST_FRAMES	2000004	Frames were lost multiple times.
ECAT_ERR_CABLE_BROKEN	2000007	The cable is broken.
ECAT_ERR_WAIT_ACK_TIMEOUT	2001000	Firmware timeout waiting for cyclic interrupt ACK.

- **Event Callback**

When the Real-Time EtherCAT MDevice Core detects an event, it triggers an event interrupt and passes a 32-bit event code to the EtherCAT MDevice Library. If the user has registered an event callback, the system will invoke that callback to inform the user of the specific event.

The event codes supported by the Event Callback are as follows:

Definition	Code	Description
ECAT_EVT_STATE_CHANGED	1000001	The EtherCAT state of the MDevice has changed.
ECAT_EVT_CABLE_RECONNECTED	1000002	The cable has been reconnected.

Functions:

- [attachCyclicCallback\(\)](#)
- [detachCyclicCallback\(\)](#)
- [attachErrorCallback\(\)](#)
- [detachErrorCallback\(\)](#)
- [attachEventCallback\(\)](#)
- [detachEventCallback\(\)](#)
- [errGetCableBrokenLocation1\(\)](#)
- [errGetCableBrokenLocation2\(\)](#)
- [evtGetMasterState\(\)](#)

## Restrictions

These callback functions are called by an interrupt handler and thus run in *Interrupt Context*. Therefore, the relevant restrictions of the interrupt context must be obeyed.

- Avoid calling **blocking** functions.
- Avoid calling **non-reentrant** functions.
- Avoid calling **time-consuming** functions.
- Avoid calling **non-interrupt-safe** functions.
  - `malloc()` / `free()`
  - `printf()` / `scanf()`
  - `fopen()` / `fclose()` / `fprintf()` / `fscanf()` / `fwrite()` / `fread()`
  - ...
- Avoid calling functions from **third-party** libraries, unless you are certain it doesn't belong to the preceding function categories.

## Floating-Point Arithmetic

FPU (Floating-Point Unit) is a specialized hardware component designed to perform floating-point arithmetic. It is typically integrated into the CPU of a computer. FPU registers are storage units within the FPU used to temporarily store data and intermediate results for floating-point arithmetic, enabling the FPU to perform complex floating-point arithmetic quickly.

When an interrupt occurs, the CPU switches from user mode to kernel mode. The FPU state at this time may differ from that required by the interrupt handler. Without saving the state, the interrupt handler may inadvertently modify the FPU state used by the user program, leading to unpredictable results. After the interrupt handler finishes executing, the original FPU state needs to be restored to guarantee that the user program can continue to execute correctly without errors caused by changes in the FPU state.

If the callback function in Interrupt Context requires floating-point arithmetic, the FPU state needs to be saved before entering the callback function, and restored after exiting the callback function to prevent affecting floating-point arithmetic of the program in Process Context.

A callback function that involves saving and restoring the FPU state is termed an FPU-enabled callback function. Conversely, a callback function that does not involve saving and restoring the FPU state is termed an FPU-disabled callback function. For details on how to register either an FPU-enabled or FPU-disabled callback function, please refer to the description of the following functions.

- [attachCyclicCallback\(\)](#)
- [attachErrorCallback\(\)](#)
- [attachEventCallback\(\)](#)

It's worth noting that saving and restoring FPU state can introduce some performance overhead, so a trade-off between performance and reliability is necessary.

## attachCyclicCallback()

### Description

Register a cyclic callback.

### Syntax

```
int attachCyclicCallback(void (*callback)(void), bool use_fpu = false);
```

### Parameters

- `[in] callback`  
The cyclic callback function to be registered, which has no parameters and no return value.
- `[in] use_fpu`  
Declare whether this callback function is FPU-enabled.
  - `true`: FPU-enabled.
  - `false`: FPU-disabled
 The default is `false`.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#) and before [EthercatMaster::start\(\)](#), or after a successful execution of [EthercatMaster::stop\(\)](#) and before [EthercatMaster::end\(\)](#).

About the usage restrictions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

void setup() {
  Serial.begin(115200);

  master.begin();
  master.attachCyclicCallback(CyclicCallback);

  // ...
}
```

```
void loop() {  
  
}
```

## detachCyclicCallback()

### Description

Unregister a cyclic callback.

### Syntax

```
int detachCyclicCallback();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#) and before [EthercatMaster::start\(\)](#), or after a successful execution of [EthercatMaster::stop\(\)](#) and before [EthercatMaster::end\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void CyclicCallback() {
  // ...
}

void setup() {
  Serial.begin(115200);

  master.begin();
  master.attachCyclicCallback(CyclicCallback);
  // ...
  master.detachCyclicCallback();
  master.end();
}

void loop() {
}
```

## attachErrorCallback()

### Description

Register an error callback.

### Syntax

```
void attachErrorCallback(void (*callback)(uint32_t), bool use_fpu = false);
```

### Parameters

- `[in] callback`  
The error callback function to be registered, which only has one parameter, a 32-bit error code.
- `[in] use_fpu`  
Declare whether this callback function is FPU-enabled.
  - `true`: FPU-enabled.
  - `false`: FPU-disabled
 The default is `false`.

### Return Value

None.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#). About the usage restrictions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"
EthercatMaster master;

void ErrorCallback(uint32_t errorcode) {
  // ...
}

void setup() {
  Serial.begin(115200);
  master.attachErrorCallback(ErrorCallback);
  master.begin();
  // ...
}

void loop() {
}
```

## detachErrorCallback()

### Description

Unregister an error callback.

### Syntax

```
void detachErrorCallback();
```

### Parameters

None.

### Returns

None.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void ErrorCallback(uint32_t errorcode) {
    // ...
}

void setup() {
    Serial.begin(115200);

    master.attachErrorCallback(ErrorCallback);
    master.begin();
    // ...
    master.end();
    master.detachErrorCallback();
}

void loop() {
}
```

## attachEventCallback()

### Description

Register an event callback.

### Syntax

```
void attachEventCallback(void (*callback)(uint32_t), bool use_fpu = false);
```

### Parameters

- `[in] callback`  
The event callback function to be registered, which only has one parameter, a 32-bit event code.
- `[in] use_fpu`  
Declare whether this callback function is FPU-enabled.
  - `true`: FPU-enabled.
  - `false`: FPU-disabled
 The default is `false`.

### Return Value

None.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#). About the usage restrictions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void EventCallback(uint32_t eventcode) {
  // ...
}

void setup() {
  Serial.begin(115200);
  master.attachEventCallback(EventCallback);
  master.begin();
  // ...
}

void loop() {
}
```

## detachEventCallback()

### Description

Unregister an event callback.

### Syntax

```
void detachEventCallback();
```

### Parameters

None.

### Return Value

None.

### Comment

This function must be called before [EthercatMaster::begin\(\)](#) or after [EthercatMaster::end\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void EventCallback(uint32_t eventcode) {
    // ...
}

void setup() {
    Serial.begin(115200);

    master.attachEventCallback(EventCallback);
    master.begin();

    // ...

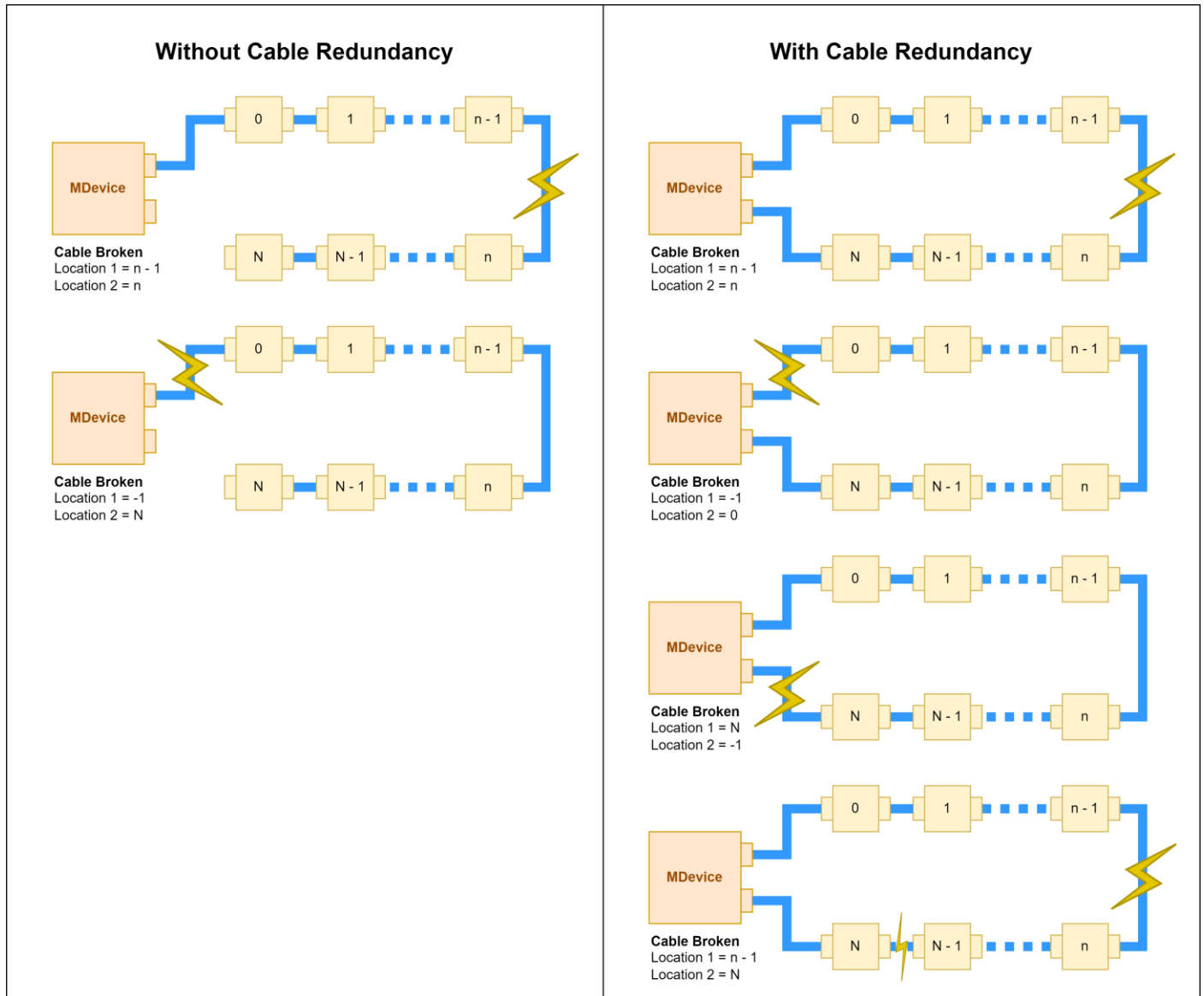
    master.end();
    master.detachEventCallback();
}

void loop() {
}
```

## errGetCableBrokenLocation1()

### Description

Get the content of the ECAT\_ERR\_CABLE\_BROKEN error, which represents the Cable Broken Location 1. For the definition of Cable Broken Location 1, see the figure below.



### Syntax

```
int errGetCableBrokenLocation1();
```

### Parameters

None.

### Return Value

Return the cable broken location 1.

### Comment

This function must be called in error callback.

### Example Code

```
#include "Ethercat.h"
```

```

EthercatMaster master;

bool CableBrokenLatched = false;
int CableBrokenLocation1;
int CableBrokenLocation2;

void ErrorCallback(uint32_t errorcode) {
    if (errorcode == ECAT_ERR_CABLE_BROKEN) {
        if (CableBrokenLatched == false) {
            CableBrokenLatched = true;
            CableBrokenLocation1 = master.errGetCableBrokenLocation1();
            CableBrokenLocation2 = master.errGetCableBrokenLocation2();
        }
    }
}

void setup() {
    Serial.begin(115200);
    master.attachErrorCallback(ErrorCallback);
    master.begin();
    master.start();
}

void loop() {
    if (CableBrokenLatched == true) {
        Serial.print("Cable broken between ");
        if (CableBrokenLocation1 < 0) Serial.print("Primary Port");
        else {
            Serial.print("SubDevice ");
            Serial.print(CableBrokenLocation1);
        }

        Serial.print(" and ");
        if (CableBrokenLocation2 < 0) Serial.println("Secondary Port");
        else {
            Serial.print("SubDevice ");
            Serial.println(CableBrokenLocation2);
        }
        CableBrokenLatched = false;
    }
}

```

## errGetCableBrokenLocation2()

### Description

Get the content of the ECAT\_ERR\_CABLE\_BROKEN error, which represents the Cable Broken Location 2. For the definition of Cable Broken Location 2, please refer to [errGetCableBrokenLocation1\(\)](#).

### Syntax

```
int errGetCableBrokenLocation2();
```

### Parameters

None.

### Return Value

Return the cable broken location 2.

### Comment

This function must be called in error callback.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

bool CableBrokenLatched = false;
int CableBrokenLocation1;
int CableBrokenLocation2;

void ErrorCallback(uint32_t errorcode) {
    if (errorcode == ECAT_ERR_CABLE_BROKEN) {
        if (CableBrokenLatched == false) {
            CableBrokenLatched = true;
            CableBrokenLocation1 = master.errGetCableBrokenLocation1();
            CableBrokenLocation2 = master.errGetCableBrokenLocation2();
        }
    }
}

void setup() {
    Serial.begin(115200);

    master.attachErrorCallback(ErrorCallback);
    master.begin();
    master.start();
}
```

```
void loop() {
  if (CableBrokenLatched == true) {
    Serial.print("Cable broken between ");
    if (CableBrokenLocation1 < 0) Serial.print("Primary Port");
    else {
      Serial.print("SubDevice ");
      Serial.print(CableBrokenLocation1);
    }

    Serial.print(" and ");
    if (CableBrokenLocation2 < 0) Serial.println("Secondary Port");
    else {
      Serial.print("SubDevice ");
      Serial.println(CableBrokenLocation2);
    }

    CableBrokenLatched = false;
  }
}
```

## evtGetMasterState()

### Description

Get the content of the ECAT\_EVT\_STATE\_CHANGED event, which represents the EtherCAT MDevice state.

### Syntax

```
int evtGetMasterState();
```

### Parameters

None.

### Return Value

Return the EtherCAT MDevice state.

### Comment

This function must be called in event callback.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
int CurrentMasterState = 0;

void EventCallback(uint32_t eventcode) {
    if (eventcode == ECAT_EVT_STATE_CHANGED)
        CurrentMasterState = master.evtGetMasterState();
}

void setup() {
    Serial.begin(115200);

    master.attachEventCallback(EventCallback);
    master.begin();
    master.start();
}

void loop() {
    Serial.print("CurrentMasterState: ");
    Serial.println(CurrentMasterState);
    delay(1000);
}
```

## 2.1.5 SubDevice Information Functions

This library provides functions to obtain information about EtherCAT SubDevices on the network. It includes querying the number of SubDevices on the network, retrieving a SubDevice's Vendor ID, Product Code, Alias Address by its sequence number, and reverse querying the SubDevice number using the aforementioned information.

This is used to identify the type of SubDevice and to choose the appropriate EtherCAT SubDevice class to attach.

Functions:

- [getSlaveCount\(\)](#)
- [getVendorID\(\)](#)
- [getProductCode\(\)](#)
- [getRevisionNumber\(\)](#)
- [getSerialNumber\(\)](#)
- [getAliasAddress\(\)](#)
- [getSlaveNo\(\)](#)

## getSlaveCount()

### Description

Get the number of SubDevices on the network.

### Syntax

```
uint16_t getSlaveCount();
```

### Parameters

None.

### Return Value

Return the number of SubDevices on the network. If an error occurs, it will return 0.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  Serial.print("SubDevice Count: ");
  Serial.println(master.getSlaveCount());
  // ...
}

void loop() {
}
```

## getVendorID()

### Description

Get the vendor ID of the specified device.

### Syntax

```
uint32_t getVendorID(uint16_t slave_no);
```

### Parameters

- `[in] slave_no`

The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.

### Return Value

Return the Vendor ID of the specified device. If an error occurs, it will return `UINT32_MAX` ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice ");
    Serial.print(i);
    Serial.print(" Vendor ID: ");
    Serial.println(master.getVendorID(i), HEX);
  }
  // ...
}

void loop() {
}
```

## getProductCode()

### Description

Get the product code of the specified device.

### Syntax

```
uint32_t getProductCode(uint16_t slave_no);
```

### Parameters

- `[in] slave_no`

The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.

### Return Value

Return the Product Code of the specified device.

If an error occurs, it will return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice ");
    Serial.print(i);
    Serial.print(" Product Code: ");
    Serial.println(master.getProductCode(i), HEX);
  }
}

void loop() {
}
```

## getRevisionNumber()

### Description

Get the revision number of the specified device.

### Syntax

```
uint32_t getRevisionNumber(uint16_t slave_no);
```

### Parameters

- `[in] slave_no`

The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.

### Return Value

Return the Revision Number of the specified device.

If an error occurs, it will return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice ");
    Serial.print(i);
    Serial.print(" Revision Number: ");
    Serial.println(master.getRevisionNumber(i), HEX);
  }
}

void loop() {
}
```

## getSerialNumber()

### Description

Get the serial number of the specified device.

### Syntax

```
uint32_t getSerialNumber(uint16_t slave_no);
```

### Parameters

- `[in] slave_no`

The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.

### Return Value

Return the Serial Number of the specified device.

If an error occurs, it will return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice ");
    Serial.print(i);
    Serial.print(" Serial Number: ");
    Serial.println(master.getSerialNumber(i), HEX);
  }
}

void loop() {
}
```

## getAliasAddress()

### Description

Get the alias address of the specified device.

### Syntax

```
int getAliasAddress(uint16_t slave_no);
```

### Parameters

- `[in] slave_no`

The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.

### Return Value

Return the Alias Address of the specified device. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice ");
    Serial.print(i);
    Serial.print(" Alias Address: ");
    Serial.println(master.getAliasAddress(i));
  }
}

void loop() {
}
```

## getSlaveNo()

### Description

Find the sequence number of the matching EtherCAT SubDevice on the network based on the input Alias Address, Vendor ID, Product Code, Revision Number, or Serial Number.

### Syntax

```
int getSlaveNo(uint16_t alias_addr);
```

```
int getSlaveNo(uint32_t vendor, uint32_t product);
```

```
int getSlaveNo(uint32_t vendor, uint32_t product, uint32_t revision, uint32_t serial_num);
```

```
int getSlaveNo(uint16_t alias_addr, uint32_t vendor, uint32_t product);
```

```
int getSlaveNo(uint16_t alias_addr, uint32_t vendor, uint32_t product, uint32_t revision, uint32_t serial_num);
```

### Parameters

- `[in] alias_addr`  
The Alias Address of the EtherCAT SubDevice you want to find. If the input value is 0, it indicates that this parameter should not be used for the search.
- `[in] vendor`  
The Vendor ID of the EtherCAT SubDevice you want to find. If the input value is 0, it indicates that this parameter should not be used for the search.
- `[in] product`  
The Product Code of the EtherCAT SubDevice you want to find. If the input value is 0, it indicates that this parameter should not be used for the search.
- `[in] revision`  
The Revision Number of the EtherCAT SubDevice you want to find. If the input value is 0, it indicates that this parameter should not be used for the search.
- `[in] serial_num`  
The Serial Number of the EtherCAT SubDevice you want to find. If the input value is 0, it indicates that this parameter should not be used for the search.

### Return Value

Return the sequence number of the matching EtherCAT SubDevice on the network. If the returned value is -1, it indicates that no matching EtherCAT SubDevice was found. Any other value less than 0 indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);

  master.begin();
  Serial.print("Search Test 1 => ");
  Serial.println(master.getSlaveNo(1200));

  Serial.print("Search Test 2 => ");
  Serial.println(master.getSlaveNo(0x00000BC3, 0x0086D324));

  Serial.print("Search Test 3 => ");
  Serial.println(master.getSlaveNo(0x00000BC3, 0x0086D304, 0x20220316,
0x00000000));

  Serial.print("Search Test 4 => ");
  Serial.println(master.getSlaveNo(251, 0x00000BC3, 0x0086D302));

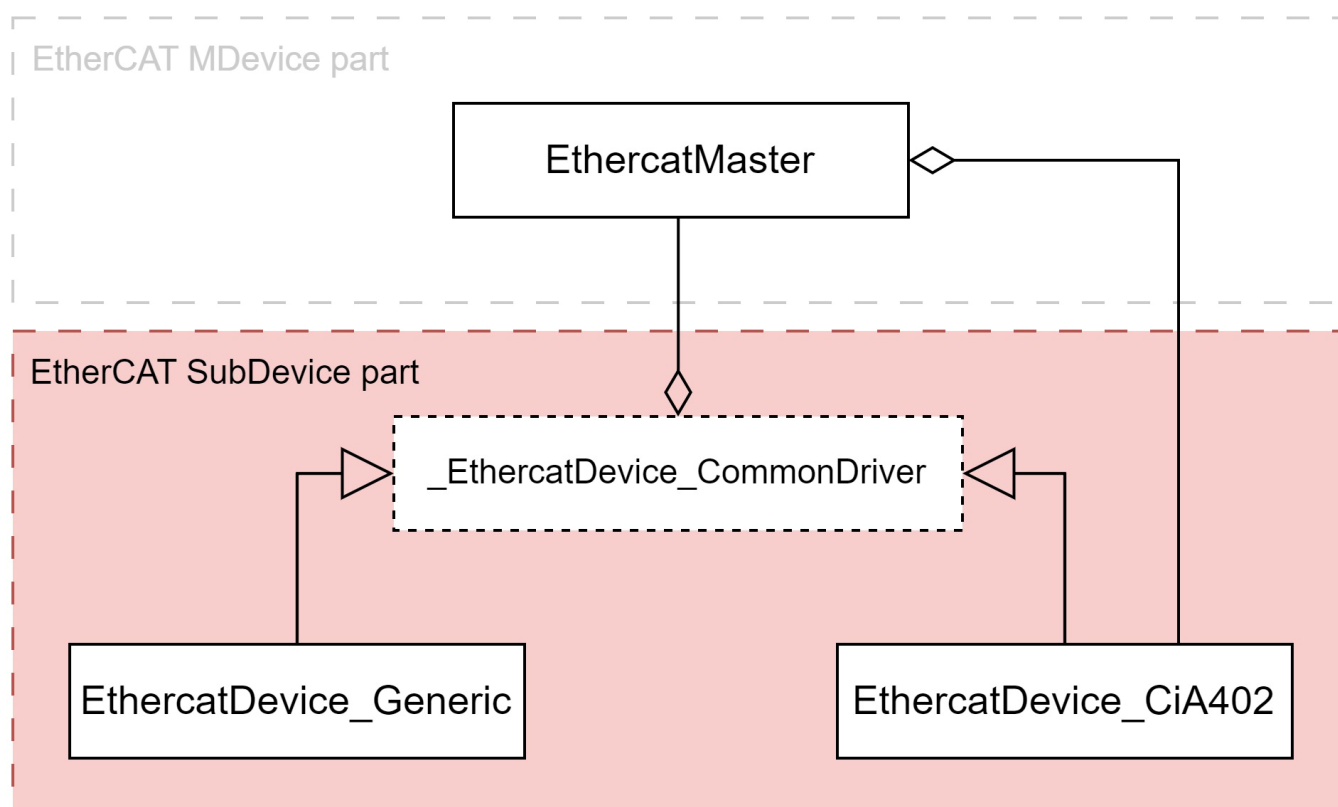
  Serial.print("Search Test 5 => ");
  Serial.println(master.getSlaveNo(252, 0x00000BC3, 0x0086D301, 0x20220316,
0x00000000));
}

void loop() {
}
```

## 2.2 EtherCAT SubDevice

The **EtherCAT SubDevice** part provides generic EtherCAT SubDevice classes, which can operate functions such as PDOs, CoE, FoE, and also includes CiA 402 SubDevice generic class.

The main class relationship between the EtherCAT SubDevice part and the EtherCAT MDevice part is association, with the EtherCAT SubDevice part depending on the EtherCAT MDevice part. As shown in the diagram below, there is an association relationship between **\_EthercatDevice\_CommonDriver** and **EthercatMaster**.



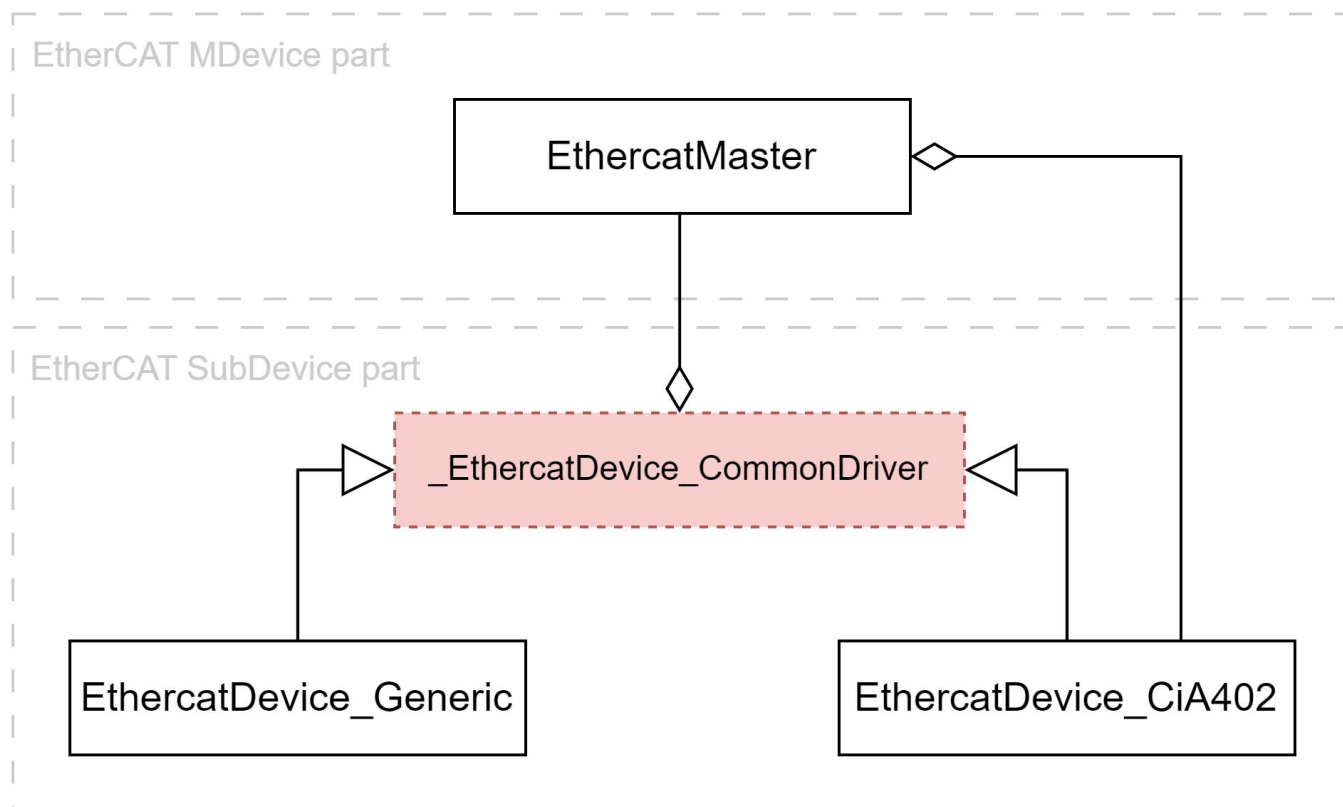
Classes:

- [\\_EthercatDevice\\_CommonDriver](#)
- [EthercatDevice\\_Generic](#)
- [EthercatDevice\\_CiA402](#)

## 2.2.1 `_EthercatDevice_CommonDriver`

`_EthercatDevice_CommonDriver` is an abstract class that not only features functions for accessing SubDevice information but also provides various EtherCAT function access methods, including PDO, SII, CoE, FoE, DC, etc. All EtherCAT SubDevice classes inherit from it.

The class relationships of `_EthercatDevice_CommonDriver` are illustrated in the following diagram:



- There is an association between `EthercatMaster` and `_EthercatDevice_CommonDriver`, with `_EthercatDevice_CommonDriver` depending on `EthercatMaster`.
- All other EtherCAT SubDevice classes inherit from `_EthercatDevice_CommonDriver`.

**WARNING:** Prohibited from declaring objects using this class.

## Functions:

Function Name	Description	Callback Available
<b>SubDevice information related functions</b>		
<a href="#">getVendorID()</a>	Get the vendor ID.	0
<a href="#">getProductCode()</a>	Get the product code.	0
<a href="#">getRevisionNumber()</a>	Get the revision number.	0
<a href="#">getSerialNumber()</a>	Get the serial number.	0
<a href="#">getAliasAddress()</a>	Get the alias address.	0
<a href="#">getSlaveNo()</a>	Get the sequence ID on the EtherCAT network.	0
<a href="#">getDeviceName()</a>	Get the device name.	0
<a href="#">getMailboxProtocol()</a>	Get the supported mailbox protocol types.	0
<a href="#">getCoEDetails()</a>	Get the details about CoE supported.	0
<a href="#">getFoEDetails()</a>	Get the details about FoE supported.	0
<a href="#">getEoEDetails()</a>	Get the details about EoE supported.	0
<a href="#">getSoEChannels()</a>	Get the number of SoE channels supported.	0
<a href="#">isSupportDC()</a>	Check if the EtherCAT SubDevice has DC supported.	0
<b>PDO access functions</b>		
<a href="#">pdoBitWrite()</a>	Write 1-bit output process data.	0
<a href="#">pdoBitRead()</a>	Read 1-bit input process data.	0
<a href="#">pdoGetOutputBuffer()</a>	Get the memory pointer of output process data.	0
<a href="#">pdoGetInputBuffer()</a>	Get the memory pointer of input process data.	0
<a href="#">pdoWrite()</a>	Write multiple bytes of output process data.	0
<a href="#">pdoWrite8()</a>	Write 8-bit output process data.	0
<a href="#">pdoWrite16()</a>	Write 16-bit output process data.	0
<a href="#">pdoWrite32()</a>	Write 32-bit output process data.	0
<a href="#">pdoWrite64()</a>	Write 64-bit output process data.	0
<a href="#">pdoRead()</a>	Read multiple bytes of input process data.	0
<a href="#">pdoRead8()</a>	Read 8-bit input process data.	0
<a href="#">pdoRead16()</a>	Read 16-bit input process data.	0
<a href="#">pdoRead32()</a>	Read 32-bit input process data.	0
<a href="#">pdoRead64()</a>	Read 64-bit input process data.	0
<b>CoE communication functions</b>		
<a href="#">sdoDownload()</a>	Write multiple bytes of data to the object.	
<a href="#">sdoDownload8()</a>	Write 8-bit value to the object.	
<a href="#">sdoDownload16()</a>	Write 16-bit value to the object.	
<a href="#">sdoDownload32()</a>	Write 32-bit value to the object.	
<a href="#">sdoDownload64()</a>	Write 64-bit value to the object.	
<a href="#">sdoUpload()</a>	Read multiple bytes of data from the object.	
<a href="#">sdoUpload8()</a>	Read 8-bit value from the object.	
<a href="#">sdoUpload16()</a>	Read 16-bit value from the object.	
<a href="#">sdoUpload32()</a>	Read 32-bit value from the object.	

<a href="#">sdoUpload64()</a>	Read 64-bit value from the object.	
<a href="#">getODlist()</a>	Get a list of objects existing in the object dictionary.	
<a href="#">getObjectDescription()</a>	Get the object description of the object.	
<a href="#">getEntryDescription()</a>	Get the entry description of the object.	
<b>FoE communication functions</b>		
<a href="#">readFoE()</a>	Read a file from the EtherCAT SubDevice.	
<a href="#">writeFoE()</a>	Write a file to the EtherCAT SubDevice.	
<b>DC configuration functions</b>		
<a href="#">setDc()</a>	Configure DC parameters.	
<b>SII EEPROM access functions</b>		
<a href="#">writeSII()</a>	Write multiple bytes of data to the SII EEPROM.	
<a href="#">writeSII8()</a>	Write 8-bit value to the SII EEPROM.	
<a href="#">writeSII16()</a>	Write 16-bit value to the SII EEPROM.	
<a href="#">writeSII32()</a>	Write 32-bit value to the SII EEPROM.	
<a href="#">readSII()</a>	Read multiple bytes of data from the SII EEPROM.	
<a href="#">readSII8()</a>	Read 8-bit value from the SII EEPROM.	
<a href="#">readSII16()</a>	Read 16-bit value from the SII EEPROM.	
<a href="#">readSII32()</a>	Read 32-bit value from the SII EEPROM.	

#### Function Groups:

- [Information](#)
- [PDO](#)
- [CoE](#)
- [FoE](#)
- [DC](#)
- [SII EEPROM](#)

## 2.2.1.1 Information Functions

The library provides functions for obtaining information about EtherCAT SubDevices on the network. This includes essential details such as Vendor ID, Product Code, Alias Address, and Device Name, used for device identification. Moreover, it offers information on whether the EtherCAT SubDevice supports specific features like CoE, FoE, DC, etc.

This SubDevice information enables users to understand the characteristics and capabilities of devices within the network and perform corresponding configuration and control tasks.

Functions:

- [getVendorID\(\)](#)
- [getProductCode\(\)](#)
- [getRevisionNumber\(\)](#)
- [getSerialNumber\(\)](#)
- [getAliasAddress\(\)](#)
- [getSlaveNo\(\)](#)
- [getDeviceName\(\)](#)
- [getMailboxProtocol\(\)](#)
- [getCoEDetails\(\)](#)
- [getFoEDetails\(\)](#)
- [getEoEDetails\(\)](#)
- [getSoEChannels\(\)](#)
- [isSupportDC\(\)](#)

## getVendorID()

### Description

Get the vendor ID of the EtherCAT SubDevice.

### Syntax

```
uint32_t getVendorID();
```

### Parameters

None.

### Return Value

Return the vendor ID of the EtherCAT SubDevice. If there is an error, return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Vendor ID: ");
  Serial.println(device.getVendorID(), HEX);
}

void loop() {
}
```

## getProductCode()

### Description

Get the product code of the EtherCAT SubDevice.

### Syntax

```
uint32_t getProductCode();
```

### Parameters

None.

### Return Value

Return the product code of the EtherCAT SubDevice.

If there is an error, return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Product Code: ");
  Serial.println(device.getProductCode(), HEX);
}

void loop() {
}
```

## getRevisionNumber()

### Description

Get the revision number of the EtherCAT SubDevice.

### Syntax

```
uint32_t getRevisionNumber();
```

### Parameters

None.

### Return Value

Return the revision number of the EtherCAT SubDevice.

If there is an error, return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Revision Number: ");
  Serial.println(device.getRevisionNumber(), HEX);
}

void loop() {
}
```

## getSerialNumber()

### Description

Get the serial number of the EtherCAT SubDevice.

### Syntax

```
uint32_t getSerialNumber();
```

### Parameters

None.

### Return Value

Return the serial number of the EtherCAT SubDevice.

If there is an error, return UINT32\_MAX ( $2^{32} - 1$ ).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Serial Number: ");
  Serial.println(device.getSerialNumber(), HEX);
}

void loop() {
}
```

## getAliasAddress()

### Description

Get the alias address of the EtherCAT SubDevice.

### Syntax

```
int getAliasAddress();
```

### Parameters

None.

### Return Value

Return the alias address of the EtherCAT SubDevice. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Alias Address: ");
  Serial.println(device.getAliasAddress());
}

void loop() {
}
```

## getSlaveNo()

### Description

Get the sequence ID of the EtherCAT SubDevice on the EtherCAT network.

### Syntax

```
int getSlaveNo();
```

### Parameters

None.

### Return Value

Return the sequence ID of the EtherCAT SubDevice on the EtherCAT network. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Sequence ID: ");
  Serial.println(device.getSlaveNo());
}

void loop() {
}
```

## getDeviceName()

### Description

Get the device name of the EtherCAT SubDevice.

### Syntax

```
char *getDeviceName(char *name, size_t len);
```

### Parameters

- [out] name  
The buffer used to store the device name.
- [in] len  
The size of the buffer used to store the device name.

### Return Value

Return the pointer to the buffer used to store the device name. If the returned value is `NULL`, it indicates an error.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);
  char name[64];

  master.begin();
  device.attach(0, master);
  Serial.print("Device Name: ");
  Serial.println(device.getDeviceName(name, 64));
}

void loop() {
}
```

## getMailboxProtocol()

### Description

Get the supported mailbox protocol types by the EtherCAT SubDevice.

### Syntax

```
int getMailboxProtocol();
```

### Parameters

None.

### Return Value

Return the supported mailbox protocol types.

- Bit 0: ADS over EtherCAT.
- Bit 1: Ethernet over EtherCAT.
- Bit 2: CAN application protocol over EtherCAT.
- Bit 3: File Access over EtherCAT.
- Bit 4: Servo Drive Profile over EtherCAT.
- Bit 5: Vendor specific protocol over EtherCAT.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Mailbox Protocols: ");
  Serial.print(device.getMailboxProtocol(), HEX);
}

void loop() {
}
```

## getCoEDetails()

### Description

Get the details about the CAN application protocol over EtherCAT (CoE) supported by the EtherCAT SubDevice.

### Syntax

```
int getCoEDetails();
```

### Parameters

None.

### Return Value

Return the details about CoE supported of the EtherCAT SubDevice.

- Bit 0: Enable SDO.
- Bit 1: Enable SDO Info.
- Bit 2: Enable PDO Assign.
- Bit 3: Enable PDO Configuration.
- Bit 4: Enable PDO Upload at startup.
- Bit 5: Enable SDO complete access.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("CoE Details: ");
  Serial.print(device.getCoEDetails(), HEX);
}

void loop() {
}
```

## getFoEDetails()

### Description

Get the details about the File Access over EtherCAT (FoE) supported by the EtherCAT SubDevice.

### Syntax

```
int getFoEDetails();
```

### Parameters

None.

### Return Value

Return the details about FoE supported of the EtherCAT SubDevice.

- Bit 0: Enable FoE.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("FoE Details: ");
  Serial.print(device.getFoEDetails(), HEX);
}

void loop() {
}
```

## getEoEDetails()

### Description

Get the details about the Ethernet over EtherCAT (EoE) supported by the EtherCAT SubDevice.

### Syntax

```
int getEoEDetails();
```

### Parameters

None.

### Return Value

Return the details about EoE supported of the EtherCAT SubDevice.

- Bit 0: Enable EoE.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("EoE Details: ");
  Serial.print(device.getEoEDetails(), HEX);
}

void loop() {
}
```

## getSoEChannels()

### Description

Get the number of Servo Drive Profile over EtherCAT (SoE) channels supported by the EtherCAT SubDevice.

### Syntax

```
int getSoEChannels();
```

### Parameters

None.

### Return Value

Return the number of SoE channels supported by the EtherCAT SubDevice. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("SoE Channels: ");
  Serial.print(device.getSoEChannels(), HEX);
}

void loop() {
}
```

## isSupportDC()

### Description

Check if the EtherCAT SubDevice has Distributed Clocks (DC) supported.

### Syntax

```
int isSupportDC();
```

### Parameters

None.

### Return Value

Return whether the EtherCAT SubDevice supports DC.

A positive value indicates support, 0 indicates no support, and a negative value indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("DC supported: ");
  Serial.println(device.isSupportDC());
}

void loop() {
}
```

## 2.2.1.2 PDO Functions

Process Data refers to real-time communication data exchanged between the MDevice and SubDevices in an EtherCAT network. This data includes information used for control, monitoring, and communication purposes. The EtherCAT MDevice cyclically transmits process data to control and monitor all SubDevices, ensuring high synchronization and low latency.

The Fieldbus Memory Management Units (FMMU) in the EtherCAT SubDevice Controller (ESC) can mapping dual-port memory to logical address. All SubDevice nodes check the EtherCAT frames sent by the EtherCAT MDevice, comparing the logical address of the process data with the configured address in the FMMU. If a match is found, the output process data is transferred to dual-port memory, and the input process data is inserted into the EtherCAT frame.

Overall, process data is an essential part of EtherCAT technology and is suitable for real-time applications in robot control, CNC control, automation control, and other fields.

Functions:

- [pdoBitWrite\(\)](#)
- [pdoBitRead\(\)](#)
- [pdoGetOutputBuffer\(\)](#)
- [pdoGetInputBuffer\(\)](#)
- [pdoWrite\(\)](#)
- [pdoWrite8\(\)](#)
- [pdoWrite16\(\)](#)
- [pdoWrite32\(\)](#)
- [pdoWrite64\(\)](#)
- [pdoRead\(\)](#)
- [pdoRead8\(\)](#)
- [pdoRead16\(\)](#)
- [pdoRead32\(\)](#)
- [pdoRead64\(\)](#)

## pdoBitWrite()

### Description

Write the specified bit value to the output process data of such SubDevice.

### Syntax

```
int pdoBitWrite(uint32_t bit_offset, uint8_t value);
```

### Parameters

- `[in] bit_offset`  
The bit offset value of output process data for the EtherCAT SubDevice.
- `[in] value`  
The bit value to be written to output process data for the EtherCAT SubDevice.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoBitWrite(0, 1);
  delay(500);
  device.pdoBitWrite(0, 0);
  delay(1000);
}
```

## pdoBitRead()

### Description

Read the specified bit value from the input process data of such SubDevice.

### Syntax

```
int pdoBitRead(uint32_t bit_offset);
```

### Parameters

- `[in] bit_offset`  
The bit offset value of input process data for the EtherCAT SubDevice.

### Return Value

The specified bit value from the input process data of such SubDevice.

If the returned value is less than zero, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Bit-0 Value:");
  Serial.print(device.pdoBitRead(0));
  Serial.print(", ");

  Serial.print("Bit-7 Value:");
  Serial.println(device.pdoBitRead(7));

  delay(1000);
}
```

## pdoGetOutputBuffer()

### Description

Get the memory pointer of the output process data for the EtherCAT SubDevice.

### Syntax

```
volatile uint8_t *pdoGetOutputBuffer();
```

### Parameters

None.

### Return Value

The memory pointer of the output process data for the EtherCAT SubDevice. If the returned value is `NULL`, it indicates an error or that such SubDevice has no output process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

volatile uint8_t *pointer = NULL;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
  pointer = device.pdoGetOutputBuffer();
}

void loop() {
  while (pointer != NULL) {
    pointer[0] = 0x55;
    delay(500);
    pointer[0] = 0xAA;
    delay(1000);
  }
}
```

## pdoGetInputBuffer()

### Description

Get the memory pointer of the input process data for the EtherCAT SubDevice.

### Syntax

```
volatile uint8_t *pdoGetInputBuffer();
```

### Parameters

None.

### Return Value

The memory pointer of the input process data for the EtherCAT SubDevice. If the returned value is `NULL`, it indicates an error or that such SubDevice has no input process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

volatile uint8_t *pointer = NULL;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
  pointer = device.pdoGetInputBuffer();
}

void loop() {
  while (pointer != NULL) {
    Serial.print("Byte-0 Value: ");
    if (pointer[0] < 0x10) Serial.print("0");
    Serial.println(pointer[0], HEX);
    delay(1000);
  }
}
```

## pdoWrite()

### Description

Write the output process data of a certain size which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoWrite(uint32_t offset, void *data, uint32_t size);
```

### Parameters

- `[in] offset`  
The byte offset value of output process data for the EtherCAT SubDevice.
- `[in] data`  
The data buffer for writing output process data.
- `[in] size`  
The size of the data buffer for writing output process data.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint8_t buffer[4] = {0x00, 0x55, 0xAA, 0xFF};

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  buffer[0] = ~buffer[0];
  buffer[1] = ~buffer[1];
```

```
buffer[2] = ~buffer[2];  
buffer[3] = ~buffer[3];  
  
device.pdowrite(0, buffer, 4);  
delay(1000);  
}
```

## pdoWrite8()

### Description

Write 8-bit output process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoWrite8(uint32_t offset, uint8_t value);
```

### Parameters

- `[in] offset`  
The byte offset value of output process data for the EtherCAT SubDevice.
- `[in] value`  
The 8-bit value to be written to output process data for the EtherCAT SubDevice.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoWrite8(0, 0x55);
  delay(500);
  device.pdoWrite8(0, 0xAA);
  delay(1000);
}
```

## pdoWrite16()

### Description

Write 16-bit output process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoWrite16(uint32_t offset, uint16_t value);
```

### Parameters

- `[in] offset`  
The byte offset value of output process data for the EtherCAT SubDevice.
- `[in] value`  
The 16-bit value to be written to output process data for the EtherCAT SubDevice.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoWrite16(0, 0x5555);
  delay(500);
  device.pdoWrite16(0, 0xAAAA);
  delay(1000);
}
```

## pdoWrite32()

### Description

Write 32-bit output process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoWrite32(uint32_t offset, uint32_t value);
```

### Parameters

- `[in] offset`  
The byte offset value of output process data for the EtherCAT SubDevice.
- `[in] value`  
The 32-bit value to be written to output process data for the EtherCAT SubDevice.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoWrite32(0, 0x55555555);
  delay(500);
  device.pdoWrite32(0, 0xAAAAAAAA);
  delay(1000);
}
```

## pdoWrite64()

### Description

Write 64-bit output process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoWrite64(uint32_t offset, uint64_t value);
```

### Parameters

- `[in] offset`  
The byte offset value of output process data for the EtherCAT SubDevice.
- `[in] value`  
The 64-bit value to be written to output process data for the EtherCAT SubDevice.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoWrite64(0, 0x5555555555555555ULL);
  delay(500);
  device.pdoWrite64(0, 0xAAAAAAAAAAAAAAAAULL);
  delay(1000);
}
```

## pdoRead()

### Description

Read the input process data of a certain size which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
int pdoRead(uint32_t offset, void *data, uint32_t size);
```

### Parameters

- `[in] offset`  
The byte offset value of input process data for the EtherCAT SubDevice.
- `[out] data`  
The data buffer for reading input process data.
- `[in] size`  
The size of the data buffer for reading input process data.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint8_t buffer[4];

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  device.pdoRead(0, buffer, 4);

  Serial.print("Buffer: ");
```

```
    if (buffer[0] < 0x10) Serial.print("0"); Serial.print(buffer[0], HEX);  
Serial.print(", ");  
    if (buffer[1] < 0x10) Serial.print("0"); Serial.print(buffer[1], HEX);  
Serial.print(", ");  
    if (buffer[2] < 0x10) Serial.print("0"); Serial.print(buffer[2], HEX);  
Serial.print(", ");  
    if (buffer[3] < 0x10) Serial.print("0"); Serial.println(buffer[3], HEX);  
  
    delay(1000);  
}
```

## pdoRead8()

### Description

Read 8-bit input process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
uint8_t pdoRead8(uint32_t offset);
```

### Parameters

- `[in] offset`  
The byte offset value of input process data for the EtherCAT SubDevice.

### Return Value

Return the 8-bit input process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.pdoRead8(0));
  delay(1000);
}
```

## pdoRead16()

### Description

Read 16-bit input process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
uint16_t pdoRead16(uint32_t offset);
```

### Parameters

- `[in] offset`

The byte offset value of input process data for the EtherCAT SubDevice.

### Return Value

Return the 16-bit input process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.pdoRead16(0));
  delay(1000);
}
```

## pdoRead32()

### Description

Read 32-bit input process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
uint32_t pdoRead32(uint32_t offset);
```

### Parameters

- [in] offset

The byte offset value of input process data for the EtherCAT SubDevice.

### Return Value

Return the 32-bit input process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.pdoRead32(0));
  delay(1000);
}
```

## pdoRead64()

### Description

Read 64-bit input process data which starting from the specified offset for the EtherCAT SubDevice.

### Syntax

```
uint64_t pdoRead64(uint32_t offset);
```

### Parameters

- `[in] offset`

The byte offset value of input process data for the EtherCAT SubDevice.

### Return Value

Return the 64-bit input process data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include <stdio.h>
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
  master.start(1000000);
}

void loop() {
  uint64_t v = device.pdoRead64(0);
  char buf[32];
  sprintf(buf, "%016lX", (unsigned long long)v);

  Serial.print("Value: ");
  Serial.println(buf);

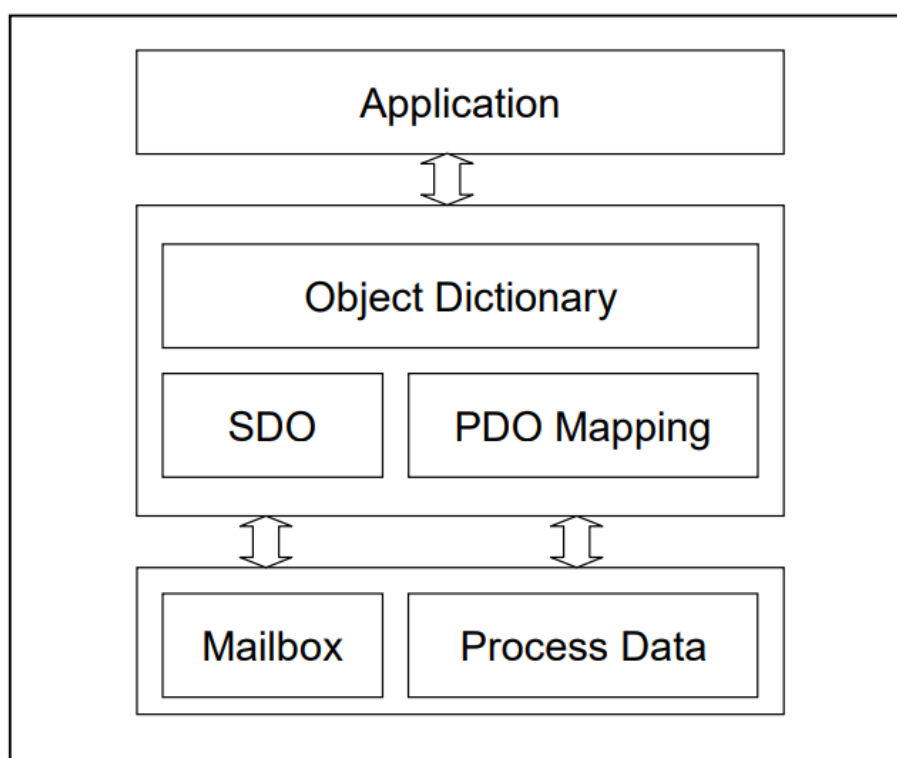
  delay(1000);
}
```

### 2.2.1.3 CoE Functions

**CANopen** is a high-level communication protocol based on the Controller Area Network (CAN) bus, commonly used for communication between control systems and devices in industrial applications. It defines a set of communication objects, data types, and network management functions to facilitate data exchange, configuration, and control between devices.

The CANopen protocol includes the following aspects:

- **Object Dictionary**  
Defines all data objects and parameters exchanged between devices. The object dictionary encompasses various types of objects such as variables, parameters, events, and functions.
- **Process Data Object (PDO)**  
Used for real-time data transmission. PDOs allow devices to transmit data between each other in a fixed or event-triggered manner, enabling real-time control and data exchange.
- **Service Data Object (SDO)**  
Used for configuring and managing device parameters. SDOs provide functionalities for reading, writing, and parameter configuration, allowing devices to dynamically exchange configuration information.



**CAN application protocol over EtherCAT (CoE)** is a CANopen protocol based on the EtherCAT network. It enables communication using the CANopen protocol over EtherCAT networks. The Object Dictionary contains parameters, application data and the mapping information between process data interface and application data (PDO mapping). Its entries can be accessed via Service Data Objects (SDO).

The SDO services primarily consist of two types of commands. The SDO command is utilized for accessing objects stored in the Object Dictionary, while the SDO information command is employed to retrieve details about these objects.

Functions:

- **SDO commands**
  - [sdoDownload\(\)](#)
  - [sdoDownload8\(\)](#)
  - [sdoDownload16\(\)](#)
  - [sdoDownload32\(\)](#)
  - [sdoDownload64\(\)](#)
  - [sdoUpload\(\)](#)
  - [sdoUpload8\(\)](#)
  - [sdoUpload16\(\)](#)
  - [sdoUpload32\(\)](#)
  - [sdoUpload64\(\)](#)
- **SDO Information commands**
  - [getODlist\(\)](#)
  - [getObjectDescription\(\)](#)
  - [getEntryDescription\(\)](#)

## sdoDownload()

### Description

Write multiple bytes of data to the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoDownload(uint16_t od_index, uint8_t od_subindex, void *data, uint32_t size, uint32_t *abortcode = NULL, bool complete_access = false, uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object. 0 or 1 if Complete Access.
- `[in] data`  
The data buffer for writing.
- `[in] size`  
The size of the data buffer for writing.
- `[out] abortcode`  
The pointer of the variable used to store the [SDO Abort Code](#). The default is `NULL`.
- `[in] complete_access`  
Use Complete Access or not. The default is `false`.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value = 0x000F;

void setup() {
  Serial.begin(115200);

  master.begin();
```

```
device.attach(0, master);  
}  
  
void loop() {  
  device.sdoDownload(0x6040, 0x00, &value, sizeof(value));  
  delay(1000);  
}
```

## sdoDownload8()

### Description

Write 8-bit value to the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoDownload8(uint16_t od_index, uint8_t od_subindex, uint8_t value,
uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object.
- `[in] value`  
The 8-bit value to be written to object.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  device.sdoDownload8(0x6040, 0x00, 0x0F);
  delay(1000);
}
```

## sdoDownload16()

### Description

Write 16-bit value to the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoDownload16(uint16_t od_index, uint8_t od_subindex, uint16_t value,
uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object.
- `[in] value`  
The 16-bit value to be written to object.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  device.sdoDownload16(0x6040, 0x00, 0x000F);
  delay(1000);
}
```

## sdoDownload32()

### Description

Write 32-bit value to the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoDownload32(uint16_t od_index, uint8_t od_subindex, uint32_t value,
uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object.
- `[in] value`  
The 32-bit value to be written to object.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
}

void loop() {
  device.sdoDownload32(0x607A, 0x00, 1000);
  delay(1000);
  device.sdoDownload32(0x607A, 0x00, 0);
  delay(1000);
}
```

## sdoDownload64()

### Description

Write 64-bit value to the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoDownload64(uint16_t od_index, uint8_t od_subindex, uint64_t value,
uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object.
- `[in] value`  
The 64-bit value to be written to object.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
}

void loop() {
  device.sdoDownload64(0x5000, 0x00, 100000);
  delay(1000);
  device.sdoDownload64(0x5000, 0x00, 0);
  delay(1000);
}
```

## sdoUpload()

### Description

Read multiple bytes of data from the specified object for the EtherCAT SubDevice.

### Syntax

```
int sdoUpload(uint16_t od_index, uint8_t od_subindex, void *data, uint32_t size, uint32_t *abortcode = NULL, bool complete_access = false, uint32_t timeout_us = 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object. 0 or 1 if Complete Access.
- `[out] data`  
The data buffer for reading.
- `[in] size`  
The size of the data buffer for reading.
- `[out] abortcode`  
The pointer of the variable used to store the [SDO Abort Code](#). The default is `NULL`.
- `[in] complete_access`  
Use Complete Access or not. The default is `false`.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value;

void setup() {
  Serial.begin(115200);

  master.begin();
```

```
device.attach(0, master);  
}  
  
void loop() {  
  device.sdoUpload(0x6040, 0x00, &value, sizeof(value));  
  Serial.print("Value: ");  
  Serial.println(value);  
  delay(1000);  
}
```

## sdoUpload8()

### Description

Read 8-bit value from the specified object for the EtherCAT SubDevice.

### Syntax

```
uint8_t sdoUpload8(uint16_t od_index, uint8_t od_subindex, uint32_t timeout_us  
= 2000000);
```

### Parameters

- `[in] od_index`  
Index of the object.
- `[in] od_subindex`  
Subindex of the object.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return the 8-bit value.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"  
  
EthercatMaster master;  
EthercatDevice_Generic device;  
  
void setup() {  
  Serial.begin(115200);  
  
  master.begin();  
  device.attach(0, master);  
}  
  
void loop() {  
  Serial.print("Value: ");  
  Serial.println(device.sdoUpload8(0x6040, 0x00));  
  delay(1000);  
}
```

## sdoUpload16()

### Description

Read 16-bit value from the specified object for the EtherCAT SubDevice.

### Syntax

```
uint16_t sdoUpload16(uint16_t od_index, uint8_t od_subindex, uint32_t
timeout_us = 2000000);
```

### Parameters

- [in] `od_index`  
Index of the object.
- [in] `od_subindex`  
Subindex of the object.
- [in] `timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return the 16-bit value.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.sdoUpload16(0x6040, 0x00));
  delay(1000);
}
```

## sdoUpload32()

### Description

Read 32-bit value from the specified object for the EtherCAT SubDevice.

### Syntax

```
uint32_t sdoUpload32(uint16_t od_index, uint8_t od_subindex, uint32_t
timeout_us = 2000000);
```

### Parameters

- [in] `od_index`  
Index of the object.
- [in] `od_subindex`  
Subindex of the object.
- [in] `timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return the 32-bit value.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.sdoUpload32(0x607A, 0x00));
  delay(1000);
}
```

## sdoUpload64()

### Description

Read 64-bit value from the specified object for the EtherCAT SubDevice.

### Syntax

```
uint64_t sdoUpload64(uint16_t od_index, uint8_t od_subindex, uint32_t
timeout_us = 2000000);
```

### Parameters

- [in] `od_index`  
Index of the object.
- [in] `od_subindex`  
Subindex of the object.
- [in] `timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return the 64-bit value.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include <stdio.h>
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  uint64_t v = device.sdoUpload64(0x5000, 0x00);
  char buf[32];

  snprintf(buf, sizeof(buf), "%016lX", (unsigned long long)v);

  Serial.print("Value: ");
  Serial.println(buf);
}
```

```
delay(1000);  
}
```

## getODlist()

### Description

Get a list of objects existing in the object dictionary for the EtherCAT SubDevice.

### Syntax

```
int getODlist(uint16_t *list, uint32_t list_size, uint32_t *abortcode = NULL,
uint32_t timeout_us = 2000000);
```

### Parameters

- `[out] list`  
The data buffer used to read the list of objects.
- `[in] list_size`  
The number of objects can be stored in the data buffer.
- `[out] abortcode`  
The pointer of the variable used to store the [SDO Abort Code](#). The default is `NULL`.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return the number of objects in the object list. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t ODlist[1024];

void setup() {
  Serial.begin(115200);

  int rc;

  master.begin();
  device.attach(0, master);

  rc = sizeof(ODlist) / sizeof(ODlist[0]);
  rc = device.getODlist(ODlist, rc);
```

```
for (int i = 0; i < rc; i++) {  
    char buf[16];  
    snprintf(buf, sizeof(buf), "0x%04X", ODlist[i]);  
    Serial.print("Index ");  
    Serial.println(buf);  
}  
  
master.end();  
}  
  
void loop() {  
}
```

## getObjectDescription()

### Description

Get the object description of the specified object addressed by index for the EtherCAT SubDevice.

### Syntax

```
int getObjectDescription(
    uint16_t    od_index,
    uint16_t *  datatype,
    uint8_t *   max_od_subindex,
    uint8_t *   objcode,
    char *      objname,
    size_t      objname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
int getObjectDescription(
    uint16_t    od_index,
    uint16_t &  datatype,
    uint8_t &   max_od_subindex,
    uint8_t &   objcode,
    char *      objname,
    size_t      objname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

### Parameters

- **[in] od\_index**  
Index of the object.
- **[out] datatype**  
The variable used to store the data type. Please refer to [Data Type](#).
- **[out] max\_od\_subindex**  
Maximum number of subindexes of the object.
- **[out] objcode**  
Object code.  
7: Variable  
8: Array  
9: Record
- **[out] objname**  
Name of the object. The buffer used to store the object name.
- **[in] objname\_size**  
The size of the buffer for the object name.

- `[out] abortcode`  
The pointer of the variable used to store the [SDO Abort Code](#). The default is `NULL`.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include <stdio.h>
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint16_t DataType;
  uint8_t MaxSubindex, ObjectCode;
  char ObjectName[64];

  master.begin();
  device.attach(0, master);

  device.getObjectDescription(0x1C12, DataType, MaxSubindex, ObjectCode,
  ObjectName, sizeof(ObjectName));

  char buf[16];

  snprintf(buf, sizeof(buf), "%04Xh", DataType);
  Serial.print("Data Type      : ");
  Serial.println(buf);

  snprintf(buf, sizeof(buf), "%02Xh", ObjectCode);
  Serial.print("Object Code    : ");
  Serial.println(buf);

  Serial.print("Max Subindex   : ");
  Serial.println(MaxSubindex);
}
```

```
Serial.print("Object Name      : ");  
Serial.println(ObjectName);  
  
master.end();  
}  
  
void loop() {  
}
```

## getEntryDescription()

### Description

Get the entry description of the specified object addressed by index and subindex for the EtherCAT SubDevice.

### Syntax

```
int getEntryDescription(
    uint16_t    od_index,
    uint8_t     od_subindex,
    uint8_t *   valueinfo,
    uint16_t *  datatype,
    uint16_t *  bitlength,
    uint16_t *  objaccess,
    char *      entryname,
    size_t      entryname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
int getEntryDescription(
    uint16_t    od_index,
    uint8_t     od_subindex,
    uint8_t &  valueinfo,
    uint16_t & datatype,
    uint16_t & bitlength,
    uint16_t & objaccess,
    char *      entryname,
    size_t      entryname_size,
    uint32_t *  abortcode = NULL,
    uint32_t    timeout_us = 2000000
);
```

### Parameters

- **[in] od\_index**  
Index of the object.
- **[in] od\_subindex**  
Subindex of the object.
- **[in] valueinfo**  
The value info includes which elements are in the response:  
Bit 0: reserved  
Bit 1: reserved  
Bit 2: reserved  
Bit 3: unit type

Bit 4: default value

Bit 5: minimum value

Bit 6: maximum value

**\*NOTE:** This parameter is only used to get the context of the **value info** in the response.

The functionality to retrieve values for unit type, default value, minimum value, and maximum value elements is not yet supported.

- `[out] datatype`  
The variable used to store the data type. Please refer to [Data Type](#).
- `[out] bitlength`  
Bit length of the object.
- `[out] objaccess`  
The attribute of access.  
Bit 0: read access in Pre-Operational state  
Bit 1: read access in Safe-Operational state  
Bit 2: read access in Operational state  
Bit 3: write access in Pre-Operational state  
Bit 4: write access in Safe-Operational state  
Bit 5: write access in Operational state  
Bit 6: object is mappable in a RxPDO  
Bit 7: object is mappable in a TxPDO  
Bit 8: object can be used for backup  
Bit 9: object can be used for settings  
Bit 10-15: reserved
- `[out] entryname`  
Name of the object entry. The buffer used to store the entry name.
- `[in] entryname_size`  
The size of the buffer for the entry name.
- `[out] abortcode`  
The pointer of the variable used to store the [SDO Abort Code](#). The default is `NULL`.
- `[in] timeout_us`  
Timeout in microseconds. The default is 2000000 us.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include <stdio.h>
#include "Ethercat.h"

EthercatMaster master;
```

```
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint16_t DataType, BitLength, ObjectAccess;
  uint8_t ValueInfo = 0;
  char EntryName[64];

  master.begin();
  device.attach(0, master);

  device.getEntryDescription(0x1C12, 0x01, ValueInfo,
                             DataType, BitLength, ObjectAccess,
                             EntryName, sizeof(EntryName));

  char buf[16];

  snprintf(buf, sizeof(buf), "%04Xh", DataType);
  Serial.print("Data Type      : ");
  Serial.println(buf);

  snprintf(buf, sizeof(buf), "%04Xh", BitLength);
  Serial.print("Bit Length    : ");
  Serial.println(buf);

  snprintf(buf, sizeof(buf), "%04Xh", ObjectAccess);
  Serial.print("Object Access  : ");
  Serial.println(buf);

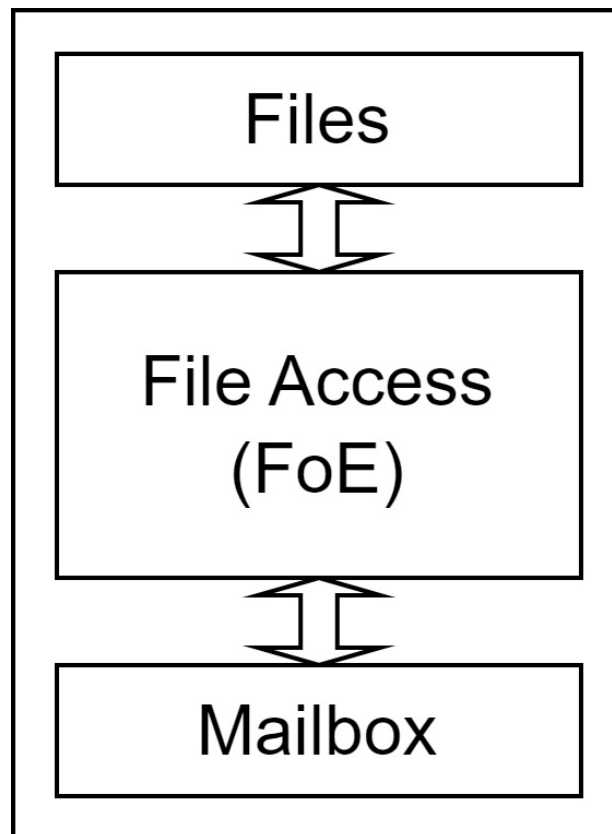
  Serial.print("Entry Name     : ");
  Serial.println(EntryName);

  master.end();
}

void loop() {
}
```

## 2.2.1.4 FoE Functions

**File access over EtherCAT (FoE)** is a protocol extension of EtherCAT that enables file transfer capabilities over the EtherCAT network. It specifies a standard way to download a firmware or any other files to the EtherCAT SubDevice or to upload a firmware or any other files from the EtherCAT SubDevice.



Functions:

- [readFoE\(\)](#)
- [writeFoE\(\)](#)

## readFoE()

### Description

Read a file from the EtherCAT SubDevice using FoE.

### Syntax

```
int readFoE(char *filename, uint32_t password, void *data, uint32_t size,
uint32_t timeout_ms = 30000);
```

### Parameters

- [in] `filename`  
Name of the file to be read.
- [in] `password`  
32-bit password value. If the password is equal to zero, it indicates that the password is unused.
- [in] `data`  
The file data buffer to be read.
- [in] `size`  
The size of the file data buffer to be read.
- [in] `timeout_ms`  
Timeout in milliseconds. The default is 30000 ms.

### Return Value

Return the size of the file to be read. If the return value is less than 0, it indicates an [error code](#).

### Comment

The function must be called after [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

#define MAX_FILE_DATA_SIZE (2 * 1024 * 1024)

EthercatMaster master;
EthercatDevice_Generic device;

char destination[] = {"destination.bin"};
char filename[] = {"firmware.bin"};
uint32_t password = 0;

uint8_t *filedata;
int filesize;
```

```
FILE *file;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  filedata = (uint8_t *)malloc(MAX_FILE_DATA_SIZE);
  if (filedata != NULL) {
    filesize = device.readFoE(filename, password, filedata,
MAX_FILE_DATA_SIZE);
    file = fopen(destination, "wb");
    if (file != NULL) {
      fwrite(filedata, sizeof(uint8_t), filesize, file);
      fclose(file);
    }
    free(filedata);
  }
}

void loop() {

}
```

## writeFoE()

### Description

Write a file to the EtherCAT SubDevice using FoE.

### Syntax

```
int writeFoE(char *filename, uint32_t password, void *data, uint32_t size,
uint32_t timeout_ms = 30000);
```

### Parameters

- `[in] filename`  
Name of the file to be written.
- `[in] password`  
32-bit password value. If the password is equal to zero, it indicates that the password is unused.
- `[in] data`  
The file data buffer to be written.
- `[in] size`  
The size of the file data buffer to be written.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 30000 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

The function must be called after [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

char source[] = {"source.bin"};
char filename[] = {"firmware.bin"};
uint32_t password = 0;

uint8_t *filedata;
int filesize;
FILE *file;

void setup() {
  Serial.begin(115200);
```

```
master.begin();
device.attach(0, master);

file = fopen(source, "rb");
if (file != NULL) {
    fseek(file, 0, SEEK_END);
    filesize = ftell(file);
    fseek(file, 0, SEEK_SET);

    filedata = (uint8_t *)malloc(filesize);
    if (filedata != NULL) {
        fread(filedata, sizeof(uint8_t), filesize, file);
        device.writeFoE(filename, password, filedata, filesize);
        free(filedata);
    }
    fclose(file);
}

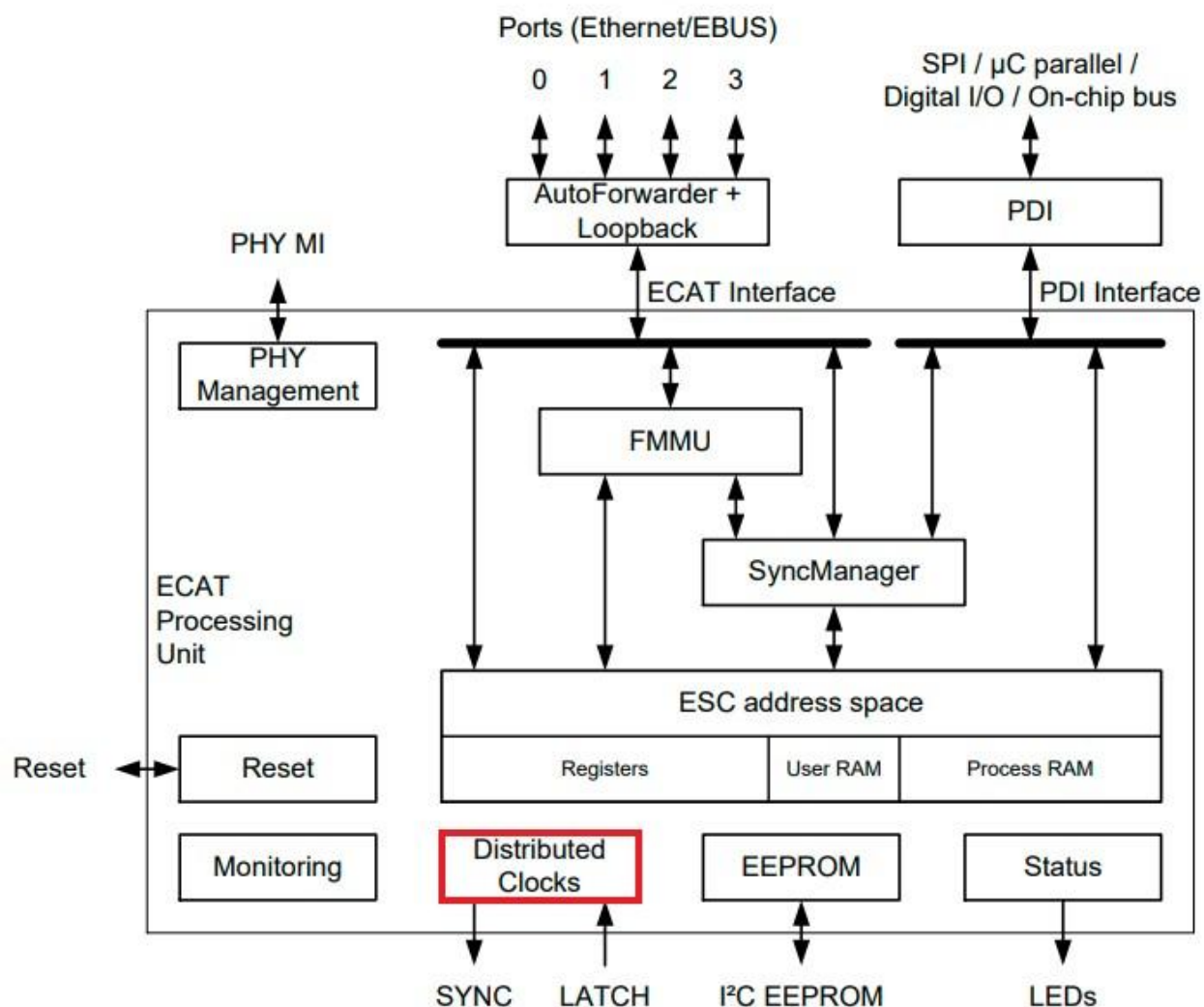
void loop() {
}
```

## 2.2.1.5 Distributed Clock (DC) Functions

The **Distributed Clocks (DC)** is an essential functional unit within the EtherCAT SubDevice Controller (ESC). It is responsible for implementing a time synchronization mechanism across the EtherCAT network, ensuring that all SubDevices synchronize their clocks according to a unified time reference, thus ensuring consistency of time across the entire system.

For system synchronization all SubDevices are synchronized to one Reference Clock. Typically, the first ESC with Distributed Clocks capability after the MDevice within one segment holds the reference time (System Time). This System Time is used as the reference clock to synchronize the DC clocks of other SubDevices and of the MDevice. The propagation delays, local clock sources drift, and local clock offsets are taken into account for the clock synchronization.

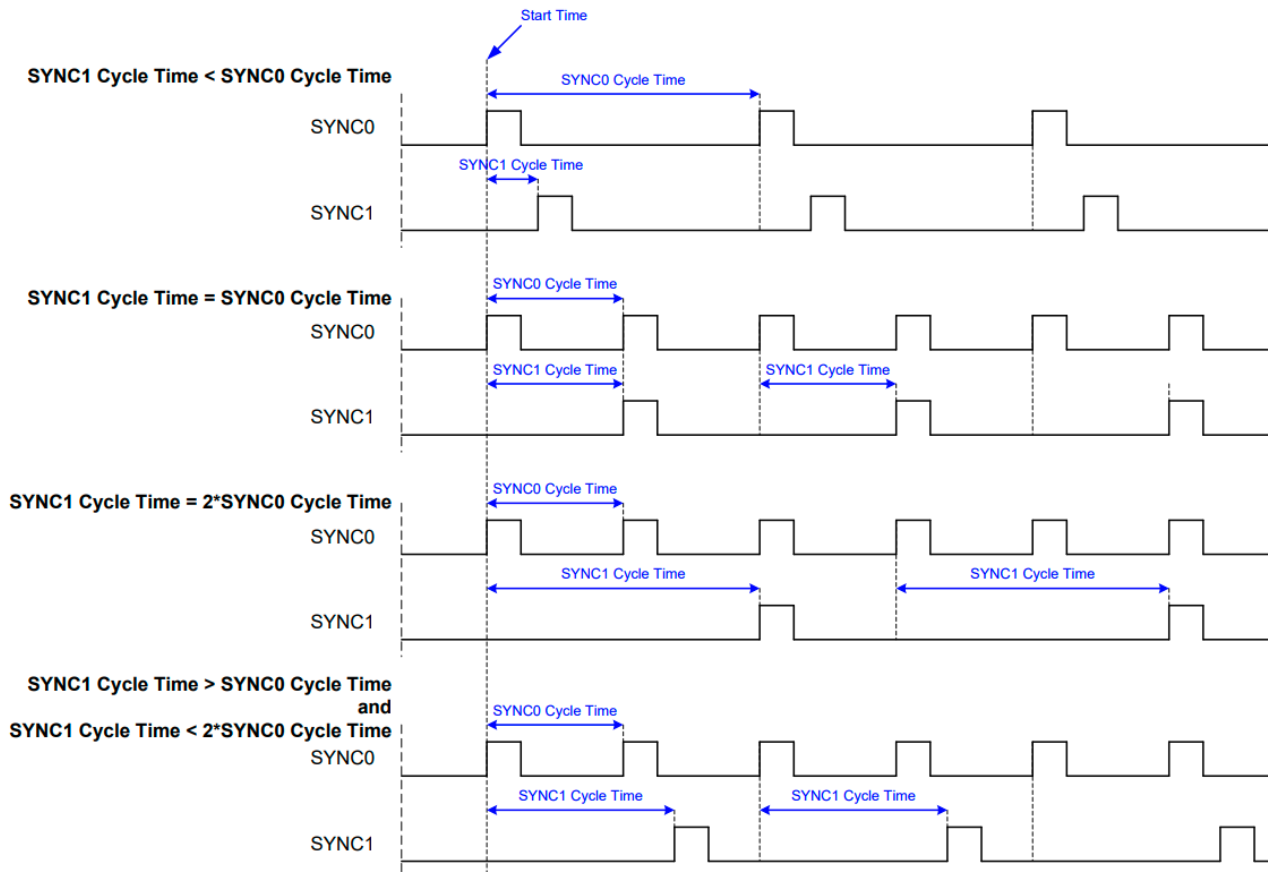
The ESCs can generate SyncSignals for local applications to be synchronized to the EtherCAT System Time. SyncSignals can be used directly (e.g., as interrupts) or for Digital Output updating/Digital Input sampling. Additionally, LatchSignals can be time stamped with respect to the EtherCAT System Time.



The DC unit supports the generation of a base SyncSignal **SYNC0** and a dependent SyncSignal **SYNC1**. The second SyncSignal (SYNC1) depends on SYNC0, it can be generated with a predefined delay after SYNC0 pulses.

If the SYNC1 Cycle Time is larger than the SYNC0 Cycle Time, it will be generated as follows: when the Start Time Cyclic Operation is reached, a SYNC0 pulse is generated. The SYNC1 pulse is generated after the SYNC0 pulse with a delay of SYNC1 Cycle Time. The next SYNC1 pulse is generated when the next SYNC0 pulse was generated, plus the SYNC1 Cycle Time.

Some example configurations are shown in the following figure:



For more detailed information, please refer to [ESC Hardware Data Sheet Section I](#).

Functions:

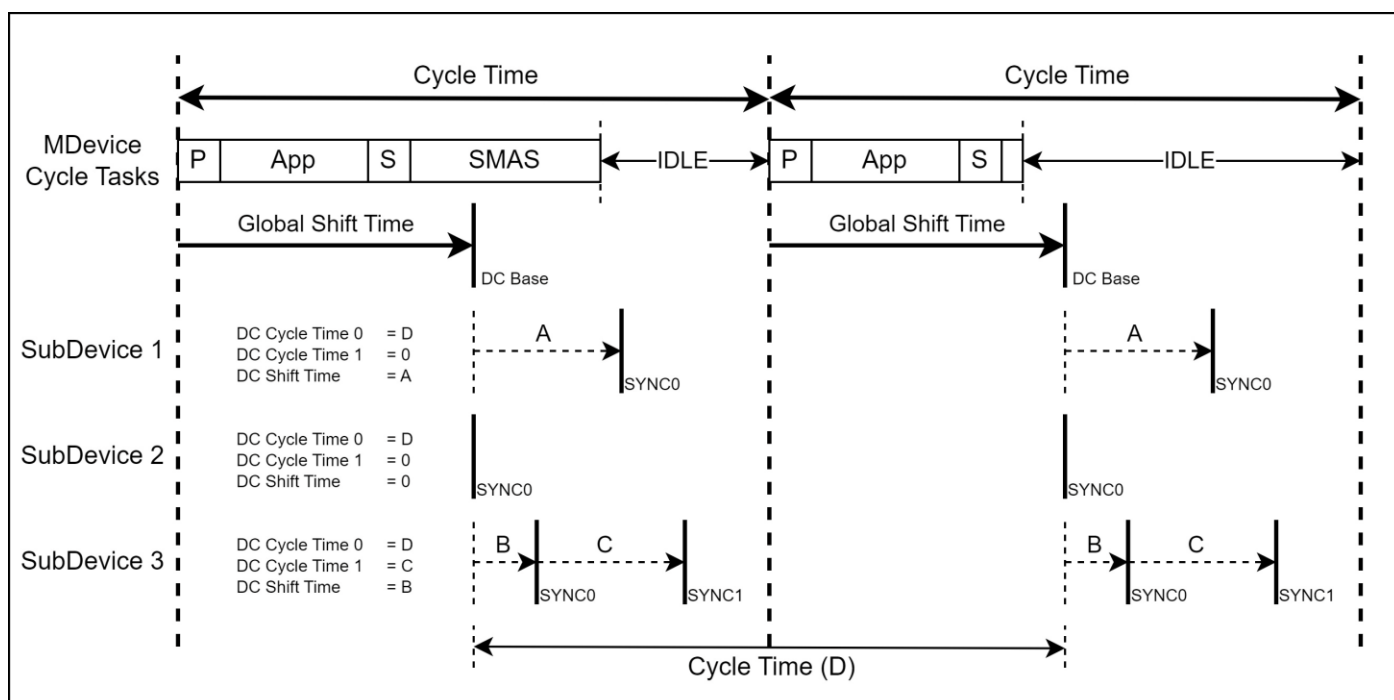
- [setDc\(\)](#)

## setDc()

**Description**

Configure DC parameters of the EtherCAT SubDevice. This function has three DC parameters to configure:

- **DC Cycle Time 0** is used to set the cycle time for the SYNC0 signal, typically aligned with the EtherCAT communication cycle time.
- **DC Cycle Time 1** is used to set the cycle time for the SYNC1 signal, which refers to the delay defined after the SYNC0 pulse. This parameter is optional.
- **DC Shift Time** is used to set the offset of the SYNC0 signal relative to the DC Base.

**Syntax**

```
int setDc(uint32_t cycletime0_ns, int32_t shifttime_ns = 0, uint32_t cycletime1_ns = 0);
```

**Parameters**

- **[in] cycletime0\_ns**  
DC SYNC0 cycle time in nanoseconds.
- **[in] shifttime\_ns**  
DC SYNC0 shift time in nanoseconds. The default is 0 ns.
- **[in] cycletime1\_ns**  
DC SYNC1 cycle time in nanoseconds. The default is 0 ns.

**Return Value**

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

## Comment

The function must be called after [EthercatMaster::begin\(\)](#) and before [EthercatMaster::start\(\)](#). This function is blocking and cannot be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.setDc(1000000);
  master.start(1000000);
}

void loop() {
  // ...
}
```

## 2.2.1.6 SII EEPROM Functions

EtherCAT SubDevice controllers use a mandatory NVRAM, typically a serial EEPROM with I<sup>2</sup>C interface, to store EtherCAT SubDevice Information (ESI). This information includes Vendor ID, Product Code, Mailbox Configuration, FMMU, PDO, and so on. EEPROM sizes from 1 Kbit up to 4 Mbit are supported, depending on the ESC.

The ESC Configuration Area (EEPROM word addresses 0 to 7) is automatically read by the ESC after power-on or reset. It contains the PDI configuration, DC settings, and the Configured Station Alias. The consistency of the ESC Configuration data is secured with a checksum. For more detailed information, please refer to [ESC Hardware Data Sheet Section I](#).

Functions:

- [readSII\(\)](#)
- [readSII8\(\)](#)
- [readSII16\(\)](#)
- [readSII32\(\)](#)
- [writeSII\(\)](#)
- [writeSII8\(\)](#)
- [writeSII16\(\)](#)
- [writeSII32\(\)](#)

## writeSII()

### Description

Write multiple bytes of data to the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
int writeSII(uint32_t offset, void *data, size_t len, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] data`  
The data buffer for writing SII EEPROM.
- `[in] len`  
The size of the data buffer for writing SII EEPROM.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint8_t buffer[4] = {0x00, 0x55, 0xAA, 0xFF};

void setup() {
  master.begin();
  device.attach(0, master);
  device.writeSII(64, buffer, 4);
}

void loop() {
}
```

## writeSII8()

### Description

Write 8-bit value to the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
int writeSII8(uint32_t offset, uint8_t value, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] value`  
The 8-bit value to be written to SII EEPROM for such EtherCAT SubDevice.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.writeSII8(64, 0x55);
}

void loop() {
}
```

## writeSII16()

### Description

Write 16-bit value to the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
int writeSII16(uint32_t offset, uint16_t value, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] value`  
The 16-bit value to be written to SII EEPROM for such EtherCAT SubDevice.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.writeSII16(64, 0x5555);
}

void loop() {
  // ...
}
```

## writeSII32()

### Description

Write 32-bit value to the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
int writeSII32(uint32_t offset, uint32_t value, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] value`  
The 32-bit value to be written to SII EEPROM for such EtherCAT SubDevice.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.writeSII32(64, 0x55555555);
}

void loop() {
  // ...
}
```

## readSII()

### Description

Read multiple bytes of data from the specified offset of SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
int readSII(uint32_t offset, void *data, size_t len, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] data`  
The data buffer for reading SII EEPROM.
- `[in] len`  
The size of the data buffer for reading SII EEPROM.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint8_t buffer[4];

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
```

```
device.readSII(0, buffer, 4);

Serial.print("Buffer: ");
Serial.print(buffer[0]); Serial.print(", ");
Serial.print(buffer[1]); Serial.print(", ");
Serial.print(buffer[2]); Serial.print(", ");
Serial.println(buffer[3]);

delay(1000);
}
```

## readSII8()

### Description

Read 8-bit value from the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
uint8_t readSII8(uint32_t offset, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return the 8-bit data of SII EEPROM.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.readSII8(0));
  delay(1000);
}
```

## readSII16()

### Description

Read 16-bit value from the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
uint16_t readSII16(uint32_t offset, uint32_t timeout_ms = 500);
```

### Parameters

- `[in] offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- `[in] timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return the 16-bit data of SII EEPROM.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.readSII16(0));
  delay(1000);
}
```

readSII32()

### Description

Read 32-bit value from the specified offset of the SII EEPROM on the EtherCAT SubDevice.

### Syntax

```
uint32_t readSII32(uint32_t offset, uint32_t timeout_ms = 500);
```

### Parameters

- [in] `offset`  
The offset value of SII EEPROM for such EtherCAT SubDevice.
- [in] `timeout_ms`  
Timeout in milliseconds. The default is 500 ms.

### Return Value

Return the 32-bit data of SII EEPROM.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

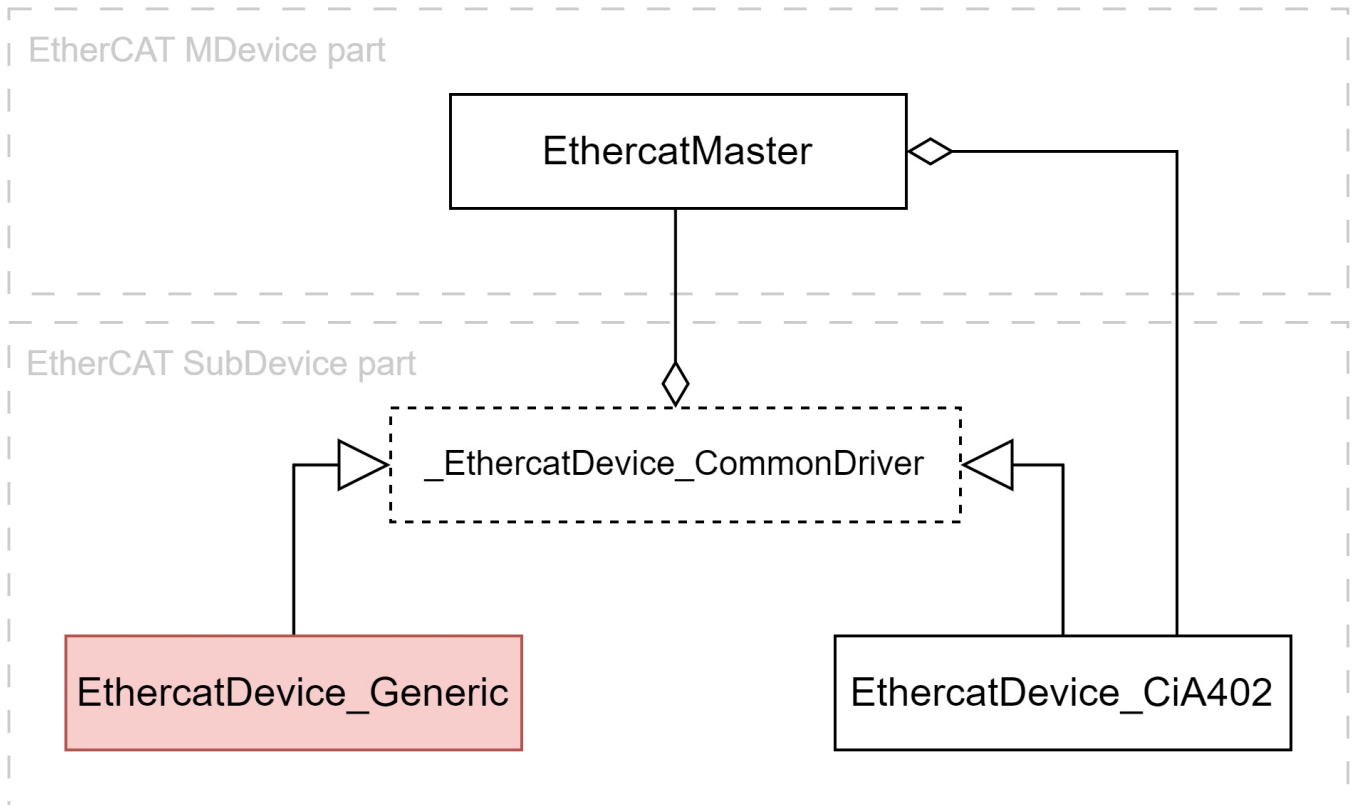
  master.begin();
  device.attach(0, master);
}

void loop() {
  Serial.print("Value: ");
  Serial.println(device.readSII32(0));
  delay(1000);
}
```

## 2.2.2 EthercatDevice\_Generic

EthercatDevice\_Generic is a generic EtherCAT SubDevice class that can be used to control all EtherCAT SubDevices, including accessing SubDevice information, PDO, CoE, FoE, DC, and more.

The class relationships of EthercatDevice\_Generic are illustrated in the following diagram:



- EthercatDevice\_Generic inherits from \_EthercatDevice\_CommonDriver.

Base Class:

- [\\_EthercatDevice\\_CommonDriver](#)

Function Groups:

- [Initialization](#)

Functions

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">attach()</a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#">detach()</a>	Deinitialize the object of this EtherCAT SubDevice class.	

## 2.2.2.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)

## attach()

### Description

Initialize the object of this EtherCAT SubDevice class and attach it to the object of EthercatMaster class based on the ID of the SubDevice on the network.

### Syntax

```
int attach(uint16_t slave_id, EthercatMaster *master, EthercatAttachMode mode = ECAT_SLAVE_NO);
```

```
int attach(uint16_t slave_id, EthercatMaster &master, EthercatAttachMode mode = ECAT_SLAVE_NO);
```

### Parameters

- **[in] slave\_id**  
The ID of the SubDevice on the EtherCAT bus. The definition of this ID is determined based on the mode parameter.
- **[in] master**  
The object of EthercatMaster class to which it should be attached.
- **[in] mode**  
The definition of slave\_id:
  - **ECAT\_SLAVE\_NO**  
The sequence number of the EtherCAT SubDevice on the network, 0 indicates the first SubDevice, 1 indicates the second SubDevice, and so on.
  - **ECAT\_ALIAS\_ADDRESS**  
The alias address of the SubDevice on the network, which is defined at byte offset 8 in the SII EEPROM of the SubDevice.

The default is **ECAT\_SLAVE\_NO**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

The function must be called after [EthercatMaster::begin\(\)](#).

\*WARNING: Prohibited from being called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
}
```

```
void loop() {  
  // ...  
}
```

## detach()

### Description

Deinitialize the object of this EtherCAT SubDevice class and detach it from the object of EthercatMaster class.

### Syntax

```
int detach();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatDevice\\_Generic::attach\(\)](#).

\*WARNING: Prohibited from being called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);

  // ...

  device.detach();
  master.end();
}

void loop() {
  // ...
}
```

## 2.2.3 EthercatDevice\_CiA402

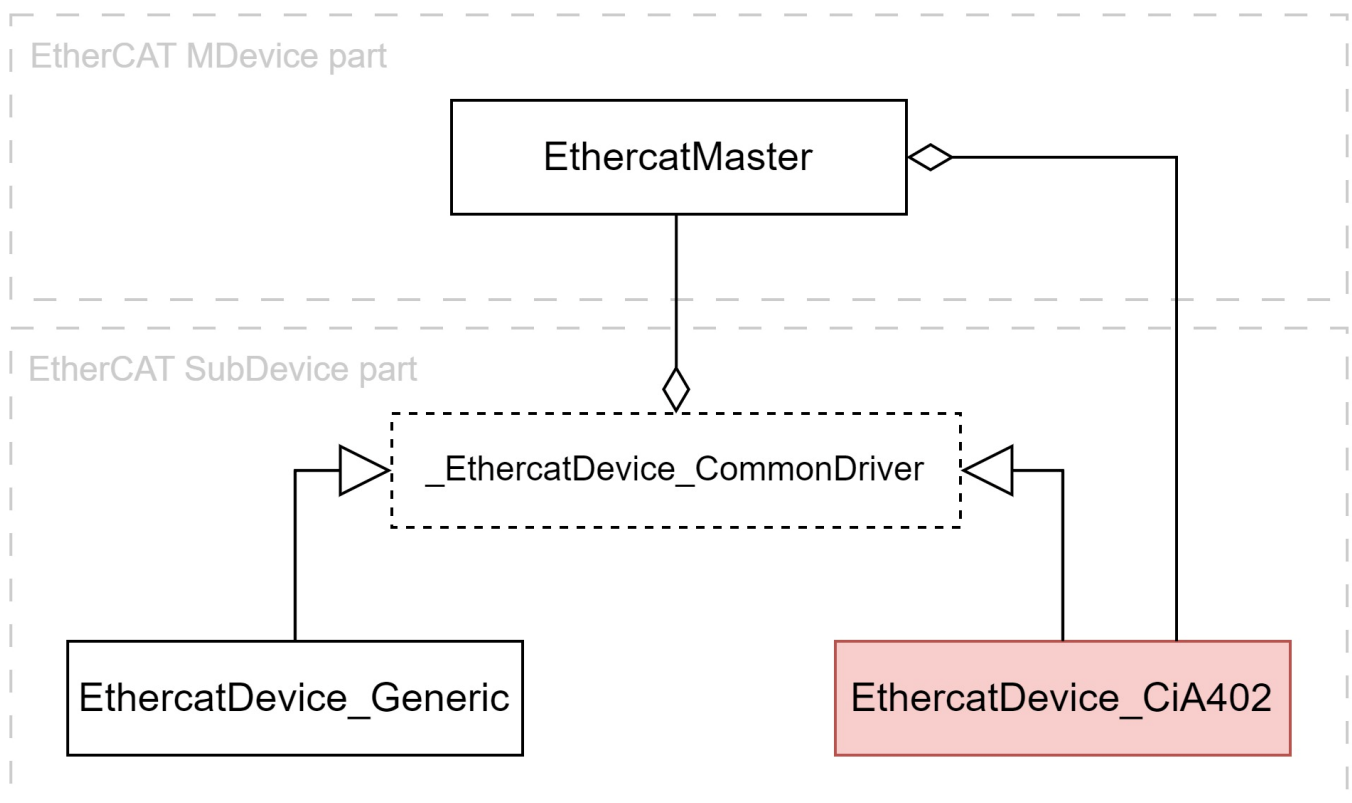
EthercatDevice\_CiA402 is a generic CiA 402 EtherCAT SubDevice class designed to control any EtherCAT servo drive that supports the CiA 402 standard. It provides access functions for commonly used CiA 402 objects and operation functions for several CiA 402 operation modes and function groups, including:

- **Operation Modes**
  - Profile Position (pp)
  - Profile Velocity (pv)
  - Profile Torque (tq)
  - Homing (hm)
  - Cyclic Synchronous Position (csp)
  - Cyclic Synchronous Velocity (csv)
  - Cyclic Synchronous Torque (cst)
- **Function Groups**
  - Touch Probe

For more detailed information about CiA 402, please refer to the following documents:

- CiA Draft Standard 402: CANopen device profile drives and motion control
- ETG.6010 Implementation Directive for CiA402 Drive Profile
- User manual for the currently used CiA 402 drive device

The class relationships of EthercatDevice\_CiA402 are illustrated in the following diagram:



- EthercatDevice\_CiA402 inherits from `_EthercatDevice_CommonDriver`.

Base Class:

- [EthercatDevice\\_CommonDriver](#)

Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
attach()	Initialize the object of this EtherCAT SubDevice class.	
detach()	Deinitialize the object of this EtherCAT SubDevice class.	
isStepper()	Check if the EtherCAT SubDevice is a stepper motor drive.	0
<b>Control-related functions</b>		
getCiA402Mode()	Get the current mode of operation. (6061 <sub>h</sub> )	0 <sup>1</sup>
setCiA402Mode()	Switch the mode of operation. (6060 <sub>h</sub> , 6061 <sub>h</sub> , 6502 <sub>h</sub> )	0 <sup>1,2</sup>
getCiA402State()	Get the current CiA 402 state. (6041 <sub>h</sub> )	0
setCiA402State()	Switch the CiA 402 state. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	0 <sup>2</sup>
enable()	Enable the drive function and power on the motor. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	0 <sup>2</sup>
disable()	Disable the drive function and power off the motor. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	0 <sup>2</sup>
<b>Operation-related functions</b>		
setTargetPosition()	Set the target position. (607A <sub>h</sub> )	0 <sup>1</sup>
setTargetVelocity()	Set the target velocity. (60FF <sub>h</sub> )	0 <sup>1</sup>
setTargetTorque()	Set the target torque. (6071 <sub>h</sub> )	0 <sup>1</sup>
setProfileAcceleration()	Set the profile acceleration. (6083 <sub>h</sub> )	0 <sup>1</sup>
setProfileDeceleration()	Set the profile deceleration. (6084 <sub>h</sub> )	0 <sup>1</sup>
setMaxAcceleration()	Set the max acceleration. (60C5 <sub>h</sub> )	0 <sup>1</sup>
setMaxDeceleration()	Set the max deceleration. (60C6 <sub>h</sub> )	0 <sup>1</sup>
setMaxProfileVelocity()	Set the max profile velocity. (607F <sub>h</sub> )	0 <sup>1</sup>
setMotionProfileType()	Set the motion profile type. (6086 <sub>h</sub> )	0 <sup>1</sup>
setPositionWindow()	Set the position window. (6067 <sub>h</sub> )	0 <sup>1</sup>
setPositionWindowTime()	Set the position window time. (6068 <sub>h</sub> )	0 <sup>1</sup>
setPositionOffset()	Set the position offset. (60B0 <sub>h</sub> )	0 <sup>1</sup>
setSoftwarePositionLimit()	Set the software position limit. (607D <sub>h</sub> )	0 <sup>1</sup>
setFollowingErrorWindow()	Set the following error window. (6065 <sub>h</sub> )	0 <sup>1</sup>
setPositionPolarity()	Set the position polarity. (607E <sub>h</sub> )	0 <sup>1</sup>
setVelocityWindow()	Set the velocity window. (606D <sub>h</sub> )	0 <sup>1</sup>
setVelocityWindowTime()	Set the velocity window time. (606E <sub>h</sub> )	0 <sup>1</sup>
setVelocityThreshold()	Set the velocity threshold. (606F <sub>h</sub> )	0 <sup>1</sup>
setVelocityOffset()	Set the velocity offset. (60B1 <sub>h</sub> )	0 <sup>1</sup>
setMaxMotorSpeed()	Set the max motor speed. (6080 <sub>h</sub> )	0 <sup>1</sup>

setVelocityPolarity()	Set the velocity polarity. (607E <sub>h</sub> )	0 <sup>1</sup>
setTorqueOffset()	Set the torque offset. (60B2 <sub>h</sub> )	0 <sup>1</sup>
setMaxTorque()	Set the max torque. (6072 <sub>h</sub> )	0 <sup>1</sup>
setPositiveTorqueLimit()	Set the positive torque limit. (60E0 <sub>h</sub> )	0 <sup>1</sup>
setNegativeTorqueLimit()	Set the negative torque limit. (60E1 <sub>h</sub> )	0 <sup>1</sup>
setQuickStopDeceleration()	Set the quick stop deceleration. (6085 <sub>h</sub> )	0 <sup>1</sup>
setQuickStopOptionCode()	Set the quick stop option code. (605A <sub>h</sub> )	
setShutdownOptionCode()	Set the shutdown option code. (605B <sub>h</sub> )	
setDisableOperationOptionCode()	Set the disable operation option code. (605C <sub>h</sub> )	
setHaltOptionCode()	Set the halt option code. (605D <sub>h</sub> )	
setFaultReactionOptionCode()	Set the fault reaction option code. (605E <sub>h</sub> )	
getErrorCode()	Get the error code. (603F <sub>h</sub> )	0 <sup>1</sup>
getSupportedDriveModes()	Get the supported drive modes. (6502 <sub>h</sub> )	0 <sup>1</sup>
getMotorResolution()	Get the motor resolution. (60EF <sub>h</sub> )	
getPositionActualValue()	Get the position actual value. (6064 <sub>h</sub> )	0 <sup>1</sup>
getVelocityActualValue()	Get the velocity actual value. (606C <sub>h</sub> )	0 <sup>1</sup>
getTorqueActualValue()	Get the torque actual value. (6077 <sub>h</sub> )	0 <sup>1</sup>
getCurrentActualValue()	Get the current actual value. (6078 <sub>h</sub> )	0 <sup>1</sup>
getPositionDemandValue()	Get the position demand value. (6062 <sub>h</sub> )	0 <sup>1</sup>
getPositionDemandInternalValue()	Get the position demand internal value. (60FC <sub>h</sub> )	0 <sup>1</sup>
getPositionActualInternalValue()	Get the position actual internal value. (6063 <sub>h</sub> )	0 <sup>1</sup>
getAdditionalPositionActualValue()	Get the additional position actual value. (60E4 <sub>h</sub> )	0 <sup>1</sup>
getFollowingErrorActualValue()	Get the following error actual value. (60F4 <sub>h</sub> )	0 <sup>1</sup>
getVelocityDemandValue()	Get the velocity demand value. (606B <sub>h</sub> )	0 <sup>1</sup>
getTorqueDemandValue()	Get the torque demand value. (6074 <sub>h</sub> )	0 <sup>1</sup>
getDigitalInputs()	Get the digital inputs. (60FD <sub>h</sub> )	0 <sup>1</sup>
<b>Profile Position mode (pp) related functions</b>		
pp_SetVelocity()	Set the profile velocity. (6081 <sub>h</sub> )	
pp_SetAcceleration()	Set the profile acceleration. (6083 <sub>h</sub> )	
pp_SetDeceleration()	Set the profile deceleration. (6084 <sub>h</sub> )	
pp_SetMotionProfileType()	Set the motion profile type. (6086 <sub>h</sub> )	
pp_Run()	Move to the target position. (6040 <sub>h</sub> , 6041 <sub>h</sub> , 607A <sub>h</sub> )	
pp_IsTargetReached()	Check if the target position has been reached. (6041 <sub>h</sub> )	0
pp_CheckFollowingError()	Check if the following error occurs. (6041 <sub>h</sub> )	0
pp_Halt()	Pause the current operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
pp_Resume()	Resume the paused operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
<b>Profile Velocity mode (pv) related functions</b>		
pv_SetAcceleration()	Set the profile acceleration. (6083 <sub>h</sub> )	
pv_SetDeceleration()	Set the profile deceleration. (6084 <sub>h</sub> )	
pv_SetMotionProfileType()	Set the motion profile type. (6086 <sub>h</sub> )	
pv_Run()	Move at a target velocity continuously. (6041 <sub>h</sub> , 60FF <sub>h</sub> )	

pv_IsTargetReached()	Check if the target velocity has been reached. (6041 <sub>h</sub> )	0
pv_CheckZeroSpeed()	Check if the speed is zero. (6041 <sub>h</sub> )	0
pv_CheckMaxSlippageError()	Check if the maximum slippage error occurs. (6041 <sub>h</sub> )	0
pv_Halt()	Pause the current operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
pv_Resume()	Resume the paused operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
<b>Profile Torque mode (tq) related functions</b>		
tq_SetTorqueSlope()	Set the torque slope. (6087 <sub>h</sub> )	
tq_SetTorqueProfileType()	Set the torque profile type. (6088 <sub>h</sub> )	
tq_SetMotorRatedCurrent()	Set the motor rated current. (6075 <sub>h</sub> )	
tq_SetMotorRatedTorque()	Set the motor rated torque. (6076 <sub>h</sub> )	
tq_Run()	Drive continuously at the target torque. (6041 <sub>h</sub> , 6071 <sub>h</sub> )	
tq_IsTargetReached()	Check if the target torque has been reached. (6041 <sub>h</sub> )	0
tq_Halt()	Pause the current operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
tq_Resume()	Resume the paused operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
<b>Homing mode (hm) related functions</b>		
hm_SetHomeOffset()	Set the home offset. (607C <sub>h</sub> )	
hm_SetHomingMethod()	Set the homing method. (6098 <sub>h</sub> )	
hm_SetHomingSpeeds()	Set the homing speeds. (6099 <sub>h</sub> )	
hm_SetHomingAcceleration()	Set the homing acceleration. (609A <sub>h</sub> )	
hm_Run()	Initiate a homing operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
hm_IsAttained()	Check the status of the homing operation. (6041 <sub>h</sub> )	0
hm_Stop()	Stop the homing operation. (6040 <sub>h</sub> , 6041 <sub>h</sub> )	
<b>Function Group "Touch Probe" related functions</b>		
enableTouchProbe1()	Enable the touch probe 1. (60B8 <sub>h</sub> , 60B9 <sub>h</sub> , 60D0 <sub>h</sub> )	
enableTouchProbe2()	Enable the touch probe 2. (60B8 <sub>h</sub> , 60B9 <sub>h</sub> , 60D0 <sub>h</sub> )	
disableTouchProbe1()	Disable the touch probe 1. (60B8 <sub>h</sub> , 60B9 <sub>h</sub> )	
disableTouchProbe2()	Disable the touch probe 2. (60B8 <sub>h</sub> , 60B9 <sub>h</sub> )	
isTouchProbe1ValueReady()	Check if a positive or negative edge has occurred on the touch probe 1 signal. (60B9 <sub>h</sub> )	0 <sup>1</sup>
isTouchProbe2ValueReady()	Check if a positive or negative edge has occurred on the touch probe 2 signal. (60B9 <sub>h</sub> )	0 <sup>1</sup>
readTouchProbe1Value()	Read the touch probe position 1. (60BA <sub>h</sub> , 60BB <sub>h</sub> )	0 <sup>1</sup>
readTouchProbe2Value()	Read the touch probe position 2. (60BC <sub>h</sub> , 60BD <sub>h</sub> )	0 <sup>1</sup>
<b>Low-level functions for mode-specific flow control</b>		
setHaltBit()	Set the halt bit in the controlword. (6040 <sub>h</sub> )	0
isTargetReached()	Check if the target has reached. (6041 <sub>h</sub> )	0
setModeSpecificBit4()	Set the bit 4 in the controlword. (6040 <sub>h</sub> )	0
setModeSpecificBit5()	Set the bit 5 in the controlword. (6040 <sub>h</sub> )	0
setModeSpecificBit6()	Set the bit 6 in the controlword. (6040 <sub>h</sub> )	0
checkModeSpecificBit12()	Check the value of bit 12 in the statusword. (6041 <sub>h</sub> )	0
checkModeSpecificBit13()	Check the value of bit 13 in the statusword. (6041 <sub>h</sub> )	0

- **Note 1:** This function can be used in callback functions if the related object is mapped to PDO.
- **Note 2:** This function will ignore the timeout parameter and will not wait for the actual value to match the set value when used in a callback.

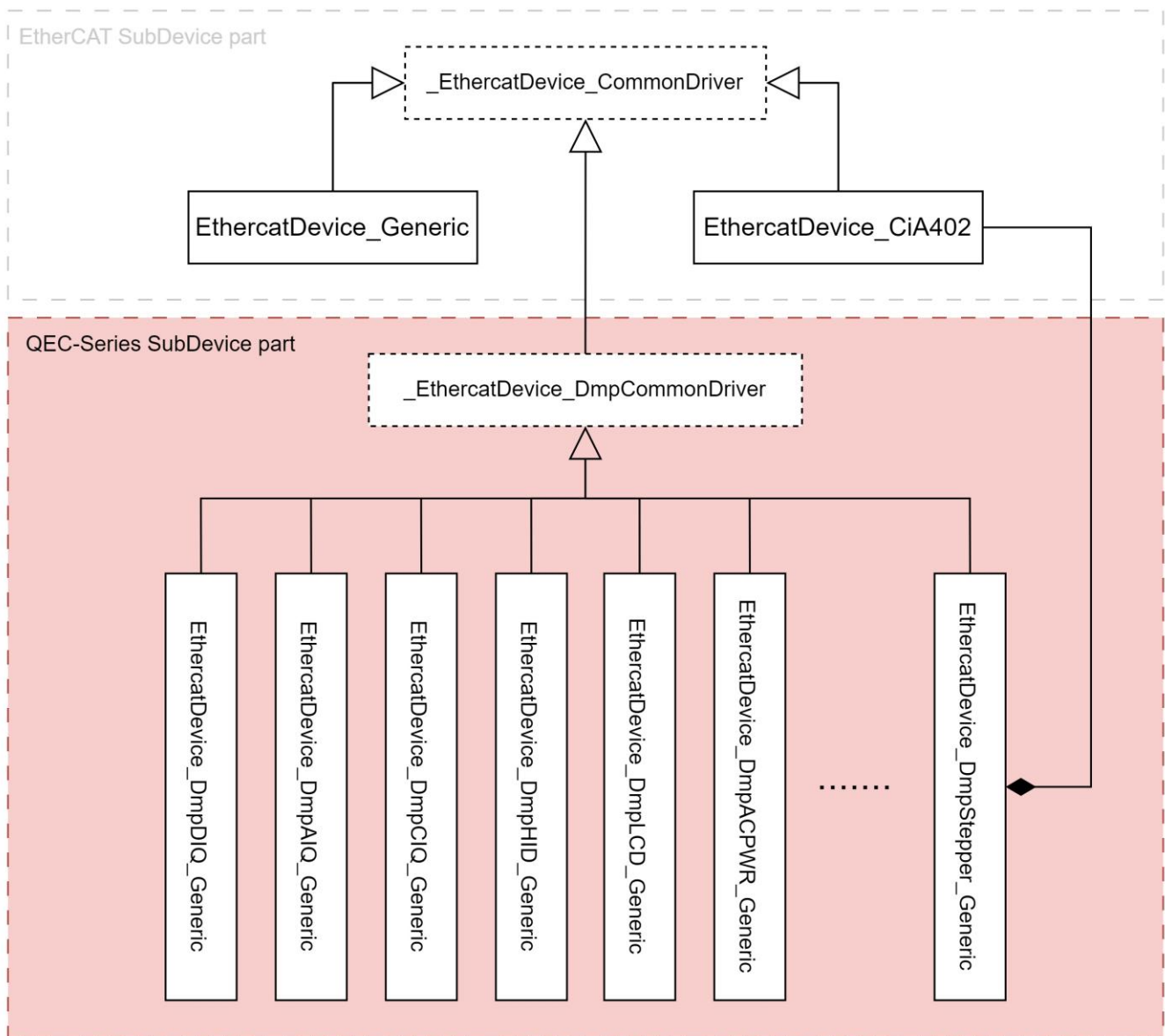
For more detailed information and API function instructions, please refer to [EtherCAT CiA402 APIs](#).

## 2.3 QEC-Series SubDevice

The QEC-Series SubDevice part provides dedicated functions for ICOP's QEC series SubDevices, enabling users to code in a more user-friendly and concise manner.

The main class relationship between the QEC-Series SubDevice part and the EtherCAT SubDevice part is association, with the QEC-Series SubDevice part depending on the EtherCAT SubDevice part.

As shown in the diagram below, there is an association relationship between `_EthercatDevice_DmpCommonDriver` and `_EthercatDevice_CommonDriver`.



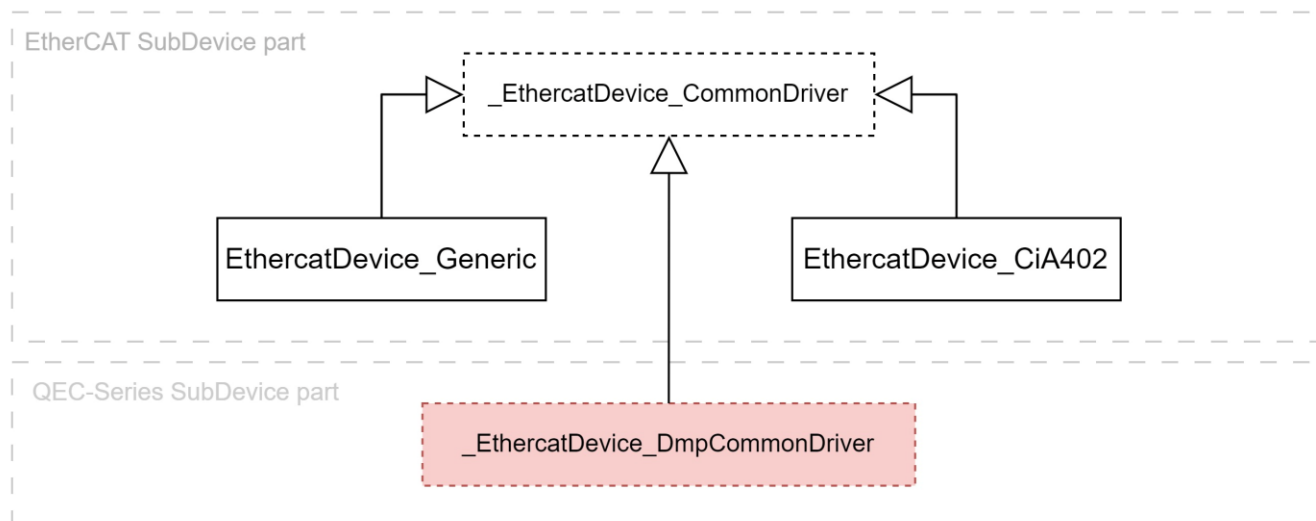
## Classes:

- `_EthercatDevice_DmpCommonDriver`
- `EthercatDevice_DmpDIQ_Generic`
- `EthercatDevice_DmpAIQ_Generic`
- `EthercatDevice_DmpCIQ_Generic`
- `EthercatDevice_DmpHID_Generic`
- `EthercatDevice_DmpLCD_Generic`
- `EthercatDevice_DmpMIC_Generic`
- `EthercatDevice_DmpVM_Generic`
- `EthercatDevice_DmpEM_Generic`
- `EthercatDevice_DmpACPWR_Generic`
- `EthercatDevice_DmpStepper_Generic`

## 2.3.1 \_EthercatDevice\_DmpCommonDriver

\_EthercatDevice\_DmpCommonDriver is an abstract class that provides dedicated access functions for EtherCAT SubDevice-specific features developed by ICOP. These functions include system monitoring (temperature, voltage, current), order information, MTBF, etc.

The class relationships of \_EthercatDevice\_DmpCommonDriver are illustrated in the following diagram:



- \_EthercatDevice\_DmpCommonDriver inherits from \_EthercatDevice\_CommonDriver.

WARNING: Prohibited from declaring objects using this class.

Base Class:

- [EthercatDevice\\_CommonDriver](#)

Functions

Function Name	Description	Callback Available
<b>System monitoring related functions</b>		
<a href="#">getSystemTemperature()</a>	Get the system temperature.	0 <sup>1,2</sup>
<a href="#">getSystemPowerVoltage()</a>	Get the system voltage.	0 <sup>1,2</sup>
<a href="#">getSystemPowerCurrent()</a>	Get the system current.	0 <sup>1,2</sup>
<a href="#">getPeripheralPowerVoltage()</a>	Get the peripheral voltage.	0 <sup>1,2</sup>
<a href="#">getPeripheralPowerCurrent()</a>	Get the peripheral current.	0 <sup>1,2</sup>
<a href="#">tryToGetSystemTemperature()</a>	Try to get the system temperature in a non-blocking manner.	0
<a href="#">tryToGetSystemPowerVoltage()</a>	Try to get the system voltage in a non-blocking manner.	0
<a href="#">tryToGetSystemPowerCurrent()</a>	Try to get the system current in a non-blocking manner.	0

<a href="#">tryToGetPeripheralPowerVoltage()</a>	Try to get the peripheral voltage in a non-blocking manner.	0
<a href="#">tryToGetPeripheralPowerCurrent()</a>	Try to get the peripheral current in a non-blocking manner.	0
<b>MTBF related functions</b>		
<a href="#">getWorkingHours()</a>	Get the working hours.	0 <sup>1</sup>
<a href="#">getBootTimes()</a>	Get the boot times.	0 <sup>1</sup>
<a href="#">tryToGetWorkingHours()</a>	Try to get the working hours in a non-blocking manner.	0
<a href="#">tryToGetBootTimes()</a>	Try to get the boot times in a non-blocking manner.	0
<b>Order information related functions</b>		
<a href="#">getCustomer()</a>	Get customer information.	
<a href="#">getOrderNumber()</a>	Get the order number.	
<a href="#">getInvoiceNumber()</a>	Get the invoice number.	
<a href="#">getDeliveryDate()</a>	Get the delivery date.	

- **Note 1:** Callback function availability is conditional and requires corresponding function usage.
- **Note 2:** This function includes floating-point arithmetic, and so cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

#### Function Groups:

- [System Monitoring](#)
- [MTBF](#)
- [Order Information](#)

## 2.3.1.1 System Monitoring Functions

The QEC series EtherCAT SubDevices with MCU all provide CoE objects to obtain system monitoring information, including system temperature, system voltage, system current, peripheral voltage, and peripheral current. Therefore, this library provides functions to get system monitoring information, enabling users to promptly monitor the system and evaluate it for any signs of failure.

Functions:

- [getSystemTemperature\(\)](#)
- [getSystemPowerVoltage\(\)](#)
- [getSystemPowerCurrent\(\)](#)
- [getPeripheralPowerVoltage\(\)](#)
- [getPeripheralPowerCurrent\(\)](#)
- [tryToGetSystemTemperature\(\)](#)
- [tryToGetSystemPowerVoltage\(\)](#)
- [tryToGetSystemPowerCurrent\(\)](#)
- [tryToGetPeripheralPowerVoltage\(\)](#)
- [tryToGetPeripheralPowerCurrent\(\)](#)

## getSystemTemperature()

### Description

Get the system temperature.

### Syntax

```
double getSystemTemperature();
```

### Parameters

None.

### Return Value

Return the system temperature.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can only be called in FPU-enabled callback functions and must be used in conjunction with [tryToGetSystemTemperature\(\)](#).

For more details about FPU-enabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("System Temperature: ");
  Serial.println(device.getSystemTemperature());
  // ...
}

void loop() {
}
```

## getSystemPowerVoltage()

### Description

Get the system voltage.

### Syntax

```
double getSystemPowerVoltage();
```

### Parameters

None.

### Return Value

Return the system voltage.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can only be called in FPU-enabled callback functions and must be used in conjunction with [tryToGetSystemPowerVoltage\(\)](#).

For more details about FPU-enabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("System Voltage: ");
  Serial.println(device.getSystemPowerVoltage());
  // ...
}

void loop() {
}
```

## getSystemPowerCurrent()

### Description

Get the system current.

### Syntax

```
double getSystemPowerCurrent();
```

### Parameters

None.

### Return Value

Return the system current.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can only be called in FPU-enabled callback functions and must be used in conjunction with [tryToGetSystemPowerCurrent\(\)](#).

For more details about FPU-enabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("System Current: ");
  Serial.println(device.getSystemPowerCurrent());
  // ...
}

void loop() {
}
```

## getPeripheralPowerVoltage()

### Description

Get the peripheral voltage.

### Syntax

```
double getPeripheralPowerVoltage();
```

### Parameters

None.

### Return Value

Return the peripheral voltage.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can only be called in FPU-enabled callback functions and must be used in conjunction with [tryToGetPeripheralPowerVoltage\(\)](#).

For more details about FPU-enabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Peripheral Voltage: ");
  Serial.println(device.getPeripheralPowerVoltage());
  // ...
}

void loop() {
}
```

## getPeripheralPowerCurrent()

### Description

Get the peripheral current.

### Syntax

```
double getPeripheralPowerCurrent();
```

### Parameters

None.

### Return Value

Return the peripheral current.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can only be called in FPU-enabled callback functions and must be used in conjunction with [tryToGetPeripheralPowerCurrent\(\)](#).

For more details about FPU-enabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Peripheral Current: ");
  Serial.println(device.getPeripheralPowerCurrent());
  // ...
}

void loop() {
}
```

## tryToGetSystemTemperature()

### Description

Try to get the system temperature in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getSystemTemperature\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetSystemTemperature();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: Completed.
- `false`: In progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
double SystemTemperature;

void CyclicCallback() {
  if (device.tryToGetSystemTemperature()) {
    SystemTemperature = device.getSystemTemperature();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  double temp = SystemTemperature;
```

```
Serial.print("System Temperature: ");  
Serial.println(temp, 2);  
}
```

## tryToGetSystemPowerVoltage()

### Description

Try to get the system voltage in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getSystemPowerVoltage\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetSystemPowerVoltage();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: Completed.
- `false`: In progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
double SystemVoltage;

void CyclicCallback() {
    if (device.tryToGetSystemPowerVoltage()) {
        SystemVoltage = device.getSystemPowerVoltage();
    }
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    double v = SystemVoltage;
```

```
Serial.print("System Voltage: ");  
Serial.println(v, 2);  
}
```

## tryToGetSystemPowerCurrent()

### Description

Try to get the system current in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getSystemPowerCurrent\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetSystemPowerCurrent();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: Completed.
- `false`: In progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
double SystemCurrent;

void CyclicCallback() {
  if (device.tryToGetSystemPowerCurrent()) {
    SystemCurrent = device.getSystemPowerCurrent();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  double cur = SystemCurrent;
```

```
Serial.print("System Current: ");  
Serial.println(cur, 2);  
}
```

## tryToGetPeripheralPowerVoltage()

### Description

Try to get the peripheral voltage in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getPeripheralPowerVoltage\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetPeripheralPowerVoltage();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: Completed.
- `false`: In progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
double PeripheralVoltage;

void CyclicCallback() {
  if (device.tryToGetPeripheralPowerVoltage()) {
    PeripheralVoltage = device.getPeripheralPowerVoltage();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  double v = PeripheralVoltage;
```

```
Serial.print("Peripheral Voltage: ");  
Serial.println(v, 2);  
}
```

## tryToGetPeripheralPowerCurrent()

### Description

Try to get the peripheral current in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getPeripheralPowerCurrent\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetPeripheralPowerCurrent();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: Completed.
- `false`: In progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
double PeripheralCurrent;

void CyclicCallback() {
  if (device.tryToGetPeripheralPowerCurrent()) {
    PeripheralCurrent = device.getPeripheralPowerCurrent();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  double cur = PeripheralCurrent;
```

```
Serial.print("Peripheral Current: ");  
Serial.println(cur, 2);  
}
```

## 2.3.1.2 MTBF Functions

MTBF stands for Mean Time Between Failures. It is a reliability metric that measures the average time between failures of a system or component. It is calculated by dividing the total time of operation by the number of failures that occur during that time. The result is an average value that can be used to estimate the expected service life of the system or component. MTBF is a useful metric for tracking the reliability of systems and components, and for identifying potential design flaws or manufacturing defects. It can also be used to make decisions about preventive maintenance schedules.

The QEC series EtherCAT SubDevices with MCU all provide CoE objects to obtain MTBF-related information. Therefore, this library provides functions to get these MTBF-related information, allowing users or users to provide it to the manufacturer to assess and judge the life and failure of the device.

Functions:

- [getWorkingHours\(\)](#)
- [getBootTimes\(\)](#)
- [tryToGetWorkingHours\(\)](#)
- [tryToGetBootTimes\(\)](#)

## getWorkingHours()

### Description

Get the current working hours.

### Syntax

```
int getWorkingHours();
```

### Parameters

None.

### Return Value

Return the current working hours. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can be called in callback functions and must be used in conjunction with [tryToGetWorkingHours\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Working Hours: ");
  Serial.println(device.getWorkingHours());
}

void loop() {
}
```

## getBootTimes()

### Description

Get the current boot times.

### Syntax

```
int getBootTimes();
```

### Parameters

None.

### Return Value

Return the current boot times. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function can be either blocking or non-blocking. If it is blocking, it cannot be called in callback functions. If it is non-blocking, it can be called in callback functions and must be used in conjunction with [tryToGetBootTimes\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Boot Times: ");
  Serial.println(device.getBootTimes());
}

void loop() {
}
```

## tryToGetWorkingHours()

### Description

Try to get the working hours in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getWorkingHours\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetWorkingHours();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: completed.
- `false`: in progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
int WorkingHours;

void CyclicCallback() {
  if (device.tryToGetWorkingHours()) {
    WorkingHours = device.getWorkingHours();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int hours = WorkingHours;
```

```
Serial.print("Working Hours: ");  
Serial.println(hours);  
}
```

## tryToGetBootTimes()

### Description

Try to get the boot times in a non-blocking manner. If it returns `true`, the reading process is complete, and you must immediately call [getBootTimes\(\)](#) to get the value. Otherwise, it indicates that the process is not yet complete.

### Syntax

```
bool tryToGetBootTimes();
```

### Parameters

None.

### Return Value

Return a Boolean value indicating whether the non-blocking read process has completed.

- `true`: completed.
- `false`: in progress.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;
int BootTimes;

void CyclicCallback() {
  if (device.tryToGetBootTimes()) {
    BootTimes = device.getBootTimes();
  }
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  int times = BootTimes;
```

```
Serial.print("Boot Times: ");  
Serial.println(times);  
}
```

### 2.3.1.3 Order Information Functions

The QEC series EtherCAT SubDevices with MCU all provide CoE objects to obtain customer order-related information, which is pre-burned into the devices before shipment.

Therefore, this library provides functions to get these customer order information, facilitating inquiries when necessary.

Functions:

- [getCustomer\(\)](#)
- [getOrderNumber\(\)](#)
- [getInvoiceNumber\(\)](#)
- [getDeliveryDate\(\)](#)

## getCustomer()

### Description

Get customer information.

### Syntax

```
char *getCustomer(char *buffer = NULL);
```

### Parameters

- `[out] buffer`  
String data buffer, please ensure that the string array size is greater than or equal to 7. If this parameter is not provided, the internal data buffer will be used.  
The default is `NULL`.

### Return Value

Return a pointer to the customer information string.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  char Customer[7];
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Customer: ");
  Serial.println(device.getCustomer());

  Serial.print("Customer: ");
  Serial.println(device.getCustomer(Customer));

  Serial.print("Customer: ");
  Serial.println(Customer);
}

void loop() {
}
```

## getOrderNumber()

### Description

Get the order number.

### Syntax

```
char *getOrderNumber(char *buffer = NULL);
```

### Parameters

- `[out] buffer`  
String data buffer, please ensure that the string array size is greater than or equal to 9. If this parameter is not provided, the internal data buffer will be used.  
The default is `NULL`.

### Return Value

Return a pointer to the order number string.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  char OrderNumber[9];
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Order Number: ");
  Serial.println(device.getOrderNumber());

  Serial.print("Order Number: ");
  Serial.println(device.getOrderNumber(OrderNumber));

  Serial.print("Order Number: ");
  Serial.println(OrderNumber);
}

void loop() {
}
```

## getInvoiceNumber()

### Description

Get the invoice number.

### Syntax

```
char *getInvoiceNumber(char *buffer = NULL);
```

### Parameters

- `[out] buffer`  
String data buffer, please ensure that the string array size is greater than or equal to 12. If this parameter is not provided, the internal data buffer will be used.  
The default is `NULL`.

### Return Value

Return a pointer to the invoice number string.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  char InvoiceNumber[12];
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Invoice Number: ");
  Serial.println(device.getInvoiceNumber());

  Serial.print("Invoice Number: ");
  Serial.println(device.getInvoiceNumber(InvoiceNumber));

  Serial.print("Invoice Number: ");
  Serial.println(InvoiceNumber);
}

void loop() {
}
```

## getDeliveryDate()

### Description

Get the delivery date.

### Syntax

```
char *getDeliveryDate(char *buffer = NULL);
```

### Parameters

- `[out] buffer`  
String data buffer, please ensure that the string array size is greater than or equal to 5. If this parameter is not provided, the internal data buffer will be used.  
The default is `NULL`.

### Return Value

Return a pointer to the delivery date string.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11DT0H device;

void setup() {
  char DeliveryDate[5];
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("Delivery Date: ");
  Serial.println(device.getDeliveryDate());

  Serial.print("Delivery Date: ");
  Serial.println(device.getDeliveryDate(DeliveryDate));

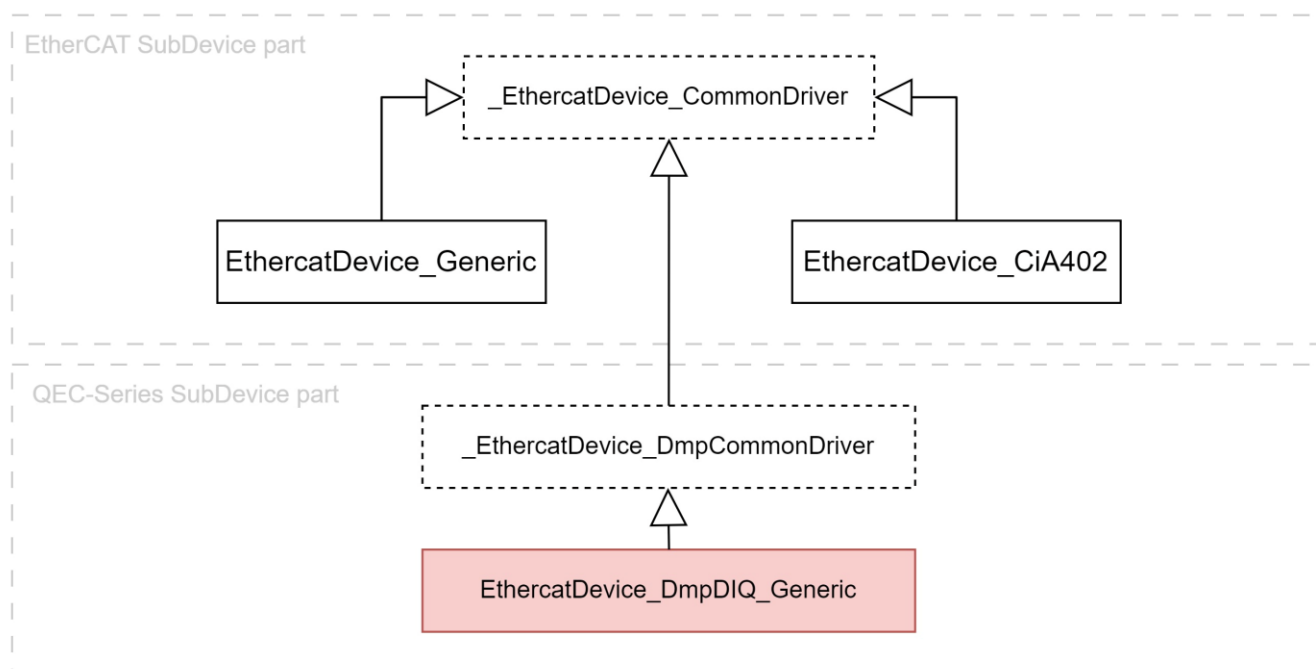
  Serial.print("Delivery Date: ");
  Serial.println(DeliveryDate);
}

void loop() {
}
```

## 2.3.2 EthercatDevice\_DmpDIQ\_Generic

EthercatDevice\_DmpDIQ\_Generic is an EtherCAT SubDevice class specifically developed by ICOP for Digital I/O EtherCAT SubDevice modules. It provides APIs for digital input, digital output, and other functionalities.

The class relationships of EthercatDevice\_DmpDIQ\_Generic are illustrated in the following diagram:



- EthercatDevice\_DmpDIQ\_Generic inherits from \_EthercatDevice\_DmpCommonDriver.

Base Class:

- [\\_EthercatDevice\\_DmpCommonDriver\(\)](#)

Derived Class:

Class Name	VID	PID	Inputs	Outputs	MCU	DC	Wire Detection
EthercatDevice_QECR00D0FS	0x00000bc3	0x0086d303	0	16	0		
EthercatDevice_QECR00D0FH	0x00000bc3	0x0086d30A	0	16	0	0	
EthercatDevice_QECR00D0TL	0x00000bc3	0x0086d327	0	32			
EthercatDevice_QECR00D0TH	0x00000bc3	0x0086d801	0	32	0	0	
EthercatDevice_QECR00DF0S	0x00000bc3	0x0086d30D	16	0	0		
EthercatDevice_QECR00DF0D	0x00000bc3	0x0086d300	16	0	0		0
EthercatDevice_QECR00DF0H	0x00000bc3	0x0086d30B	16	0	0	0	
EthercatDevice_QECR00DT0L	0x00000bc3	0x0086d323	32	0			
EthercatDevice_QECR00DT0H	0x00000bc3	0x0086d701	32	0	0	0	
EthercatDevice_QECR00D88S	0x00000bc3	0x0086d309	8	8	0		

EthercatDevice_QECR00D88D	0x00000bc3	0x0086d301	8	8	0		0
EthercatDevice_QECR00D88H	0x00000bc3	0x0086d30F	8	8	0	0	
EthercatDevice_QECR00DC4D	0x00000bc3	0x0086d304	12	4	0		0
EthercatDevice_QECR00D4CD	0x00000bc3	0x0086d302	4	12	0		0
EthercatDevice_QECR00DF0K	0x00000bc3	0x0086d311	16	0	0	0	
EthercatDevice_QECR00D0FK	0x00000bc3	0x0086d313	0	16	0	0	
EthercatDevice_QECR00D88K	0x00000bc3	0x0086d315	8	8	0	0	
EthercatDevice_QECR11D0FS	0x00000bc3	0x0086d0d4	0	16	0		
EthercatDevice_QECR11D0FH	0x00000bc3	0x0086d305	0	16	0	0	
EthercatDevice_QECR11D0TL	0x00000bc3	0x0086d324	0	32			
EthercatDevice_QECR11D0TH	0x00000bc3	0x0086d800	0	32	0	0	
EthercatDevice_QECR11DF0S	0x00000bc3	0x0086d30E	16	0	0		
EthercatDevice_QECR11DF0D	0x00000bc3	0x0086d0d2	16	0	0		0
EthercatDevice_QECR11DF0H	0x00000bc3	0x0086d306	16	0	0	0	
EthercatDevice_QECR11DT0L	0x00000bc3	0x0086d320	32	0			
EthercatDevice_QECR11DT0H	0x00000bc3	0x0086d700	32	0	0	0	
EthercatDevice_QECR11D88S	0x00000bc3	0x0086d0d5	8	8	0		
EthercatDevice_QECR11D88D	0x00000bc3	0x0086d307	8	8	0		0
EthercatDevice_QECR11D88H	0x00000bc3	0x0086d308	8	8	0	0	
EthercatDevice_QECR11DF0K	0x00000bc3	0x0086d310	16	0	0	0	
EthercatDevice_QECR11D0FK	0x00000bc3	0x0086d312	0	16	0	0	
EthercatDevice_QECR11D88K	0x00000bc3	0x0086d314	8	8	0	0	

#### Function Groups:

- [Initialization](#)
- [Digital I/O](#)
- [Broken Wire Detection](#)

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#"><u>attach()</u></a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#"><u>detach()</u></a>	Deinitialize the object of this EtherCAT SubDevice class.	
<b>Digital I/O access functions</b>		
<a href="#"><u>digitalWrite()</u></a>	Write a value to a specified digital output pin.	0
<a href="#"><u>digitalRead()</u></a>	Read the value from a specified digital input pin.	0
<a href="#"><u>digitalWriteAll()</u></a>	Write the digital output state of all pins.	0
<a href="#"><u>digitalReadAll()</u></a>	Read the digital input state of all pins.	0
<b>Broken wire detection functions</b>		
<a href="#"><u>startBrokenWireTest()</u></a>	Initiates a broken wire detection test.	

## 2.3.2.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_DmpDIQ\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)

## attach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::attach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR00D0FH.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D0FH device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // ...
}
```

## detach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::detach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR00D0FH.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D0FH device;

void setup() {
  master.begin();
  device.attach(0, master);

  delay(3000);

  device.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 2.3.2.2 Digital I/O Functions

Digital input and digital output functions for the EthercatDevice\_DmpDIQ\_Generic class.

Functions:

- [digitalWrite\(\)](#)
- [digitalRead\(\)](#)
- [digitalWriteAll\(\)](#)
- [digitalReadAll\(\)](#)

## digitalWrite()

### Description

Write a **HIGH** or a **LOW** value to a specified digital output pin on the EtherCAT SubDevice.

### Syntax

```
int digitalWrite(int pin, uint8_t value);
```

### Parameters

- **[in] pin**  
The digital output pin number of the EtherCAT SubDevice.
- **[in] value**  
Digital logic level, the value is **HIGH** or **LOW**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWrite(0, HIGH);
  delay(1000);
  device.digitalWrite(0, LOW);
  delay(1000);
}
```

## digitalRead()

### Description

Read the value from a specified digital input pin on the EtherCAT SubDevice.

### Syntax

```
int digitalRead(int pin);
```

### Parameters

- [in] pin

The digital input pin number of the EtherCAT SubDevice.

### Return Value

Return the digital logic level of the specified digital pin, the value is HIGH or LOW.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWrite(0, HIGH);
  delay(1000);
  Serial.print("DI0: ");
  Serial.println(device.digitalRead(0));

  device.digitalWrite(0, LOW);
  delay(1000);
  Serial.print("DI0: ");
  Serial.println(device.digitalRead(0));
}
```

## digitalWriteAll()

### Description

Write the digital output state of all pins on the EtherCAT SubDevice simultaneously.

### Syntax

```
int digitalWriteAll(uint32_t value);
```

### Parameters

- [in] value

This parameter is a 32-bit unsigned integer that specifies the value to be written to all digital output pins. Each bit in the value corresponds to a digital output pin of the EtherCAT SubDevice, with the following mapping:

- Bit 0 indicates digital output pin 0.
- Bit 1 indicates digital output pin 1.
- And so on, up to bit 31 which indicates digital output pin 31.

A value of 1 typically sets the corresponding pin to **HIGH**, while a value of 0 sets it to **LOW**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWriteAll(0x55555555);
  delay(1000);
  device.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
}
```

## digitalReadAll()

### Description

Read the digital input state of all pins on the EtherCAT SubDevice simultaneously.

### Syntax

```
uint32_t digitalReadAll();
```

### Parameters

None.

### Return Value

Return a 32-bit unsigned integer that represents the combined state of all digital input pins. Each bit in the returned value corresponds to a digital input pin of the EtherCAT SubDevice, with the following mapping:

- Bit 0 indicates digital input pin 0.
- Bit 1 indicates digital input pin 1.
- And so on, up to bit 31 which indicates digital input pin 31.

A value of 1 indicates that the corresponding pin is currently **HIGH**, while a value of 0 indicates that it is currently **LOW**.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWriteAll(0x55555555);
  delay(1000);
  Serial.print("DI: ");
  Serial.println(device.digitalReadAll(), HEX);
}
```

```
device.digitalWriteAll(0xAFFFFFFF);  
delay(1000);  
Serial.print("DI: ");  
Serial.println(device.digitalReadAll(), HEX);  
}
```

### 2.3.2.3 Broken Wire Detection Functions

Broken Wire Detection Functions for the EthercatDevice\_DmpDIQ\_Generic class.

Functions:

- [startBrokenWireTest\(\)](#)

## startBrokenWireTest()

### Description

Initiates a broken wire detection test on the specified input pins of the EtherCAT SubDevice.

### Syntax

```
uint16_t startBrokenWireTest(uint16_t bitMask, uint32_t timeout_ms = 1000);
```

### Parameters

- `[in] bitmask`

This parameter is an unsigned 16-bit integer that specifies which input pins to include in the broken wire detection test. Each bit in the `bitMask` corresponds to an input pin of the EtherCAT SubDevice, with the following mapping:

- Bit 0 indicates digital input pin 0.
- Bit 1 indicates digital input pin 1.
- And so on, up to bit 15 which indicates digital input pin 15

A value of 1 indicates that the corresponding pin should be included in the test, while a value of 0 indicates that it should be excluded.

**\*WARNING:** It is crucial to ensure that this parameter does not contain bits set to 1 for input channels that do not exist on the EtherCAT SubDevice, otherwise this test will not be executed.

- `[in] timeout_ms`

Timeout in milliseconds. The default is 1000 ms.

### Return Value

Return a 16-bit unsigned integer that represents the results of the broken wire detection test. Each bit in the returned value corresponds to an input pin included in the test. The interpretation of each bit is as follows:

- 0: The corresponding pin was either not included in the test or was detected as broken.
- 1: The corresponding pin was included in the test and was detected as connected and not broken.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00D88D device;

void setup() {
```

```
Serial.begin(115200);

master.begin();
device.attach(0, master);
master.start();

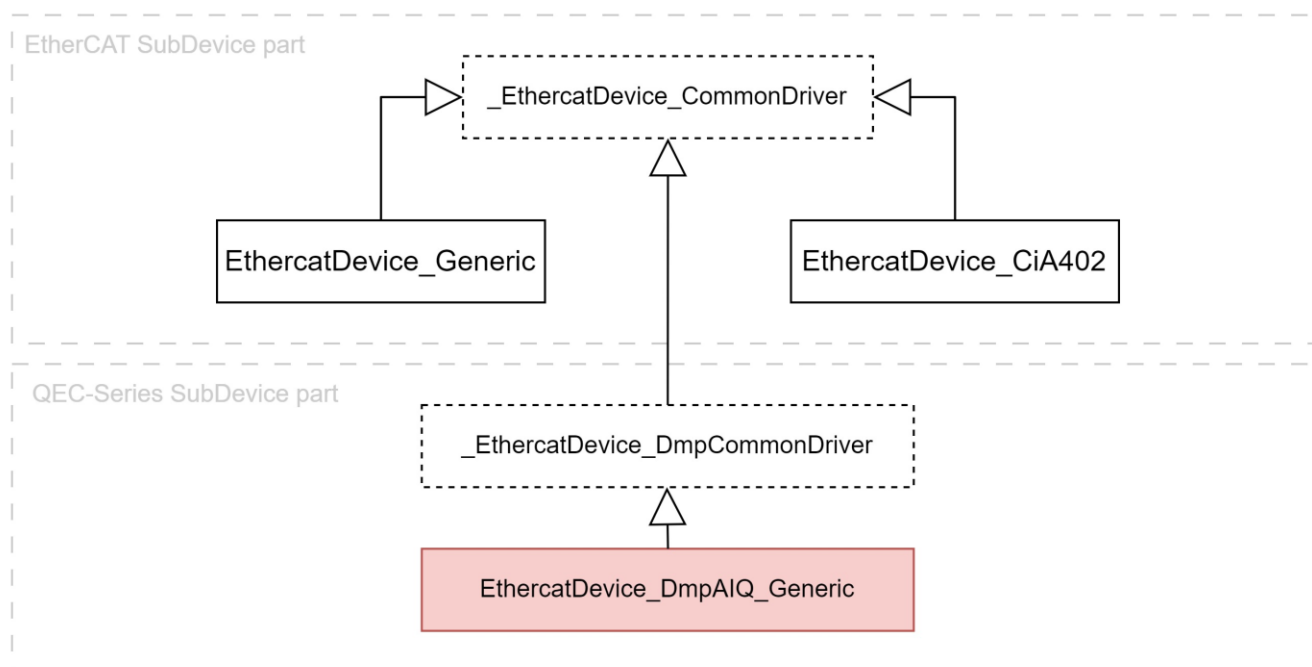
Serial.println(device.startBrokenWireTest(0xFF), HEX);
}

void loop() {
}
```

### 2.3.3 EthercatDevice\_DmpAIQ\_Generic

EthercatDevice\_DmpAIQ\_Generic is an EtherCAT SubDevice class specifically developed by ICOP for Analog I/O EtherCAT SubDevice modules. It provides APIs for analog input, analog output, and other functionalities.

The class relationships of EthercatDevice\_DmpAIQ\_Generic are illustrated in the following diagram:



- EthercatDevice\_DmpAIQ\_Generic inherits from \_EthercatDevice\_DmpCommonDriver.

Base Class:

- [\\_EthercatDevice\\_DmpCommonDriver](#)

Derived Class:

Class Name	Vendor ID	Product Code	Input Channels	Output Channels
EthercatDevice_QECR11A44S	0x00000bc3	0x0086D880	4	4
EthercatDevice_QECR11A40S	0x00000bc3	0x0086D881	4	0
EthercatDevice_QECR11A04S	0x00000bc3	0x0086D882	0	4
EthercatDevice_QECR00A44S	0x00000bc3	0x0086D883	4	4

## Function Groups:

- [Initialization](#)
- [Analog Output](#)
- [Analog Input](#)

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">attach()</a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#">detach()</a>	Deinitialize the object of this EtherCAT SubDevice class.	
<b>Analog output access functions</b>		
<a href="#">analogWrite()</a>	Write a value to the specified analog output channel.	0
<a href="#">voltageWrite()</a>	Write a voltage value to the specified analog output channel.	0 <sup>1</sup>
<a href="#">currentWrite()</a>	Write a current value to the specified analog output channel.	0 <sup>1</sup>
<b>Analog input access functions</b>		
<a href="#">analogRead()</a>	Read the value of the specified analog input channel.	0
<a href="#">voltageRead()</a>	Read the voltage value of the specified analog input channel.	0 <sup>1</sup>

- **Note 1:** This function includes floating-point arithmetic, and so cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

## 2.3.3.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_DmpAIQ\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)

## attach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::attach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11A44S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // ...
}
```

## detach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::detach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11A44S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  master.begin();
  device.attach(0, master);

  delay(3000);

  device.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 2.3.3.2 Analog Output Access Functions

Analog output access functions for the EthercatDevice\_DmpAIQ\_Generic class.

Functions:

- [analogWrite\(\)](#)
- [voltageWrite\(\)](#)
- [currentWrite\(\)](#)

## analogWrite()

### Description

Write a value to the specified analog output channel on the EtherCAT SubDevice.

### Syntax

```
int analogWrite(int ch, int32_t value);
```

### Parameters

- **[in] ch**  
The specified analog output channel on the EtherCAT SubDevice.
- **[in] value**  
The analog output value to be written. This parameter is a 32-bit signed integer whose value is linearly mapped to a physical output by the following rule, according to the mode configuration of the specified channel:
  - Voltage Mode: Maps a value of 0 to voltage 0V,  $2^{31}$  to 64V, and  $-2^{31}$  to -64V. (i.e.,  $2^{25}$  would get mapped to 1V.)
  - Current Mode: Maps a value of 0 to current 0A,  $2^{31}$  to 16A, and  $-2^{31}$  to -16A. (i.e.,  $2^{27}$  would get mapped to 1A.)

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

int voltage = 5;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.analogWrite(0, voltage << 25);
  delay(1000);
}
```

```
device.analogWrite(0, 0);  
delay(1000);  
  
device.analogWrite(0, -1 * (voltage << 25));  
delay(1000);  
  
device.analogWrite(0, 0);  
delay(1000);  
}
```

## voltageWrite()

### Description

Write a voltage value to the specified analog output channel on the EtherCAT SubDevice. This function is used to specify that the channel is configured for Voltage Mode.

### Syntax

```
int voltageWrite(int ch, double voltage);
```

### Parameters

- `[in] ch`  
The specified analog output channel on the EtherCAT SubDevice.
- `[in] voltage`  
The desired output voltage, in volts (V).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions.

For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.voltageWrite(0, 10.0);
  delay(1000);

  device.voltageWrite(0, 0);
  delay(1000);

  device.voltageWrite(0, -10.0);
  delay(1000);
}
```

```
device.voltageWrite(0, 0);  
delay(1000);  
}
```

## currentWrite()

### Description

Write a current value to the specified analog output channel on the EtherCAT SubDevice. This function is used to specify that the channel is configured for Current Mode.

### Syntax

```
int currentWrite(int ch, double current);
```

### Parameters

- `[in] ch`  
The specified analog output channel on the EtherCAT SubDevice.
- `[in] current`  
The desired output current, in amperes (A).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions.

For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.currentWrite(0, 0.02);
  delay(1000);

  device.currentWrite(0, 0);
  delay(1000);

  device.currentWrite(0, 0.01);
  delay(1000);
}
```

```
device.currentWrite(0, 0);  
delay(1000);  
}
```

### 2.3.3.3 Analog Input Access Functions

Analog input access functions for the EthercatDevice\_DmpAIQ\_Generic class.

Functions:

- [analogRead\(\)](#)
- [voltageRead\(\)](#)

## analogRead()

### Description

Read the value of the specified analog input channel on the EtherCAT SubDevice.

### Syntax

```
int analogRead(int ch);
```

### Parameters

- `[in] ch`

The specified analog input channel on the EtherCAT SubDevice.

### Return Value

Return a 32-bit signed integer within the range of  $-2^{31}$  to  $2^{31}$  as the analog input value.

This value is linearly mapped to a physical input voltage in volts, where 0 maps to 0V,  $2^{31}$  maps to 64V, and  $-2^{31}$  maps to -64V.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AI0: ");
  Serial.println(device.analogRead(0));
  delay(1000);
}
```

## voltageRead()

### Description

Read the voltage value of the specified analog input channel on the EtherCAT SubDevice.

### Syntax

```
double voltageRead(int ch);
```

### Parameters

- [in] ch

The specified analog input channel on the EtherCAT SubDevice.

### Return Value

Return the voltage value expressed in volts (V) as a double.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions.

For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11A44S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

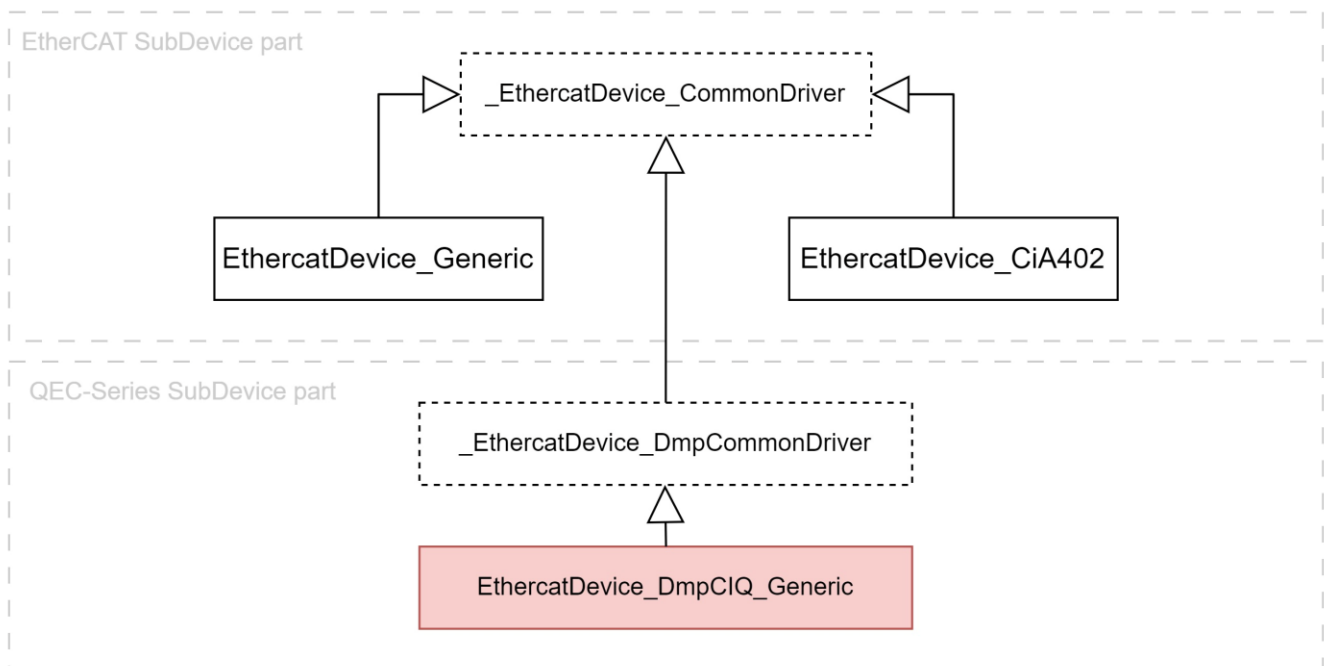
void loop() {
  Serial.print("AI0: ");
  Serial.print(device.voltageRead(0), 6);
  Serial.println(" V");
  delay(1000);
}
```

## 2.3.4 EthercatDevice\_DmpCIQ\_Generic

EthercatDevice\_DmpCIQ\_Generic is an EtherCAT SubDevice class specifically developed by ICOP for Compound I/O EtherCAT SubDevice modules. It encompasses Digital I/O, Analog I/O, and RS232/RS485 functionalities.

**RS232** and **RS485** are electrical specifications that define the voltage levels, signal timing, and connector pinouts for implementing UART communication over physical cables. **UART (Universal Asynchronous Receiver Transmitter)** is a hardware interface standard for asynchronous serial data communication. It defines the format of data bits, start and stop bits, parity bits, and baud rate for serial communication. UART is commonly used for connecting microcontrollers, computers, and other devices for data exchange. This device features one UART port that can be freely switched between RS232 or RS485 modes to accommodate user application requirements.

The class relationships of EthercatDevice\_DmpCIQ\_Generic are illustrated in the following diagram:



- EthercatDevice\_DmpCIQ\_Generic inherits from \_EthercatDevice\_DmpCommonDriver.

Base Class:

- [\\_EthercatDevice\\_DmpCommonDriver](#)

## Derived Class:

Class Name	Vendor ID	Product Code	DI	DO	AI	AO	UART
EthercatDevice_QECR00CFFG	0x00000bc3	0x0086d900	16	16	1	1	1
EthercatDevice_QECR11CFFG	0x00000bc3	0x0086d903	16	16	1	1	1
EthercatDevice_QECR00CFFU	0x00000bc3	0x0086d914	16	16	0	0	1
EthercatDevice_QECR11CFFU	0x00000bc3	0x0086d917	16	16	0	0	1
EthercatDevice_QECR00CFFL	0x00000bc3	0x0086d918	16	16	0	0	0
EthercatDevice_QECR11CFFL	0x00000bc3	0x0086d91B	16	16	0	0	0
EthercatDevice_QECR00CT0U	0x00000bc3	0x0086D91D	32	0	0	0	1
EthercatDevice_QECR11CT0U	0x00000bc3	0x0086D91C	32	0	0	0	1
EthercatDevice_QECR00C0TU	0x00000bc3	0x0086D91F	0	32	0	0	1
EthercatDevice_QECR11C0TU	0x00000bc3	0x0086D91E	0	32	0	0	1

## Function Groups:

- Initialization-related
- Control-related
- Digital I/O
- Analog I/O
- UART

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">attach()</a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#">detach()</a>	Deinitialize the object of this EtherCAT SubDevice class.	
<b>Control-related functions</b>		
<a href="#">update()</a>	Update state machines and internal variables for each function.	0
<b>Digital I/O access functions</b>		
<a href="#">digitalWrite()</a>	Write a value to a specified digital output pin.	0
<a href="#">digitalRead()</a>	Read the value from a specified digital input pin.	0
<a href="#">digitalWriteAll()</a>	Write the digital output state of all pins.	0
<a href="#">digitalReadAll()</a>	Read the digital input state of all pins.	0
<b>Analog I/O access functions</b>		
<a href="#">analogWrite()</a>	Write a value to the specified analog output channel.	0
<a href="#">voltageWrite()</a>	Write a voltage value to the specified analog output channel.	0 <sup>1</sup>
<a href="#">currentWrite()</a>	Write a current value to the specified analog output channel.	0 <sup>1</sup>
<a href="#">analogRead()</a>	Read the value of the specified analog input channel.	0
<a href="#">voltageRead()</a>	Read the voltage value of the specified analog input channel.	0 <sup>1</sup>
<b>UART access functions</b>		
<a href="#">uartIs485()</a>	Check if the specified UART port is in RS485 mode.	0
<a href="#">uartSetBaud()</a>	Configure the baud rate.	

<a href="#">uartSetFormat()</a>	Configure the UART frame format.	
<a href="#">uartSetFlowControl()</a>	Configure the flow control mode.	
<a href="#">uartGetRTS()</a>	Get the current state of the RTS control signal.	0
<a href="#">uartGetCTS()</a>	Get the current state of the CTS signal.	0
<a href="#">uartGetDTR()</a>	Get the current state of the DTR control signal.	0
<a href="#">uartGetDSR()</a>	Get the current state of the DSR signal.	0
<a href="#">uartSetRTS()</a>	Control the RTS signal.	0
<a href="#">uartSetDTR()</a>	Control the DTR signal.	0
<a href="#">uartClearFIFO()</a>	Clear the TX and RX FIFOs.	
<a href="#">uartClearTxQueue()</a>	Clear the software TX FIFO.	0
<a href="#">uartClearRxQueue()</a>	Clear the software RX FIFO.	0
<a href="#">uartQueryTxQueue()</a>	Get the current number of bytes in the software TX FIFO.	0
<a href="#">uartQueryRxQueue()</a>	Get the current number of bytes in the software RX FIFO.	0
<a href="#">uartTxQueueEmpty()</a>	Check if the software TX FIFO is empty.	0
<a href="#">uartRxQueueEmpty()</a>	Check if the software RX FIFO is empty.	0
<a href="#">uartTxQueueFull()</a>	Check if the software TX FIFO is full.	0
<a href="#">uartRxQueueFull()</a>	Check if the software RX FIFO is full.	0
<a href="#">uartSend()</a>	Transmit multiple byte data.	0
<a href="#">uartWrite()</a>	Transmit one byte data.	0
<a href="#">uartReceive()</a>	Receive multiple byte data.	0
<a href="#">uartRead()</a>	Read one byte data.	0

- **Note 1:** This function includes floating-point arithmetic, and so cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

## 2.3.4.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_DmpCIQ\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)

## attach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::attach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11CFFG.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}
void loop() {
  // ...
}
```

## detach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::detach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11CFFG.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);

  delay(3000);

  device.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 2.3.4.2 Control Functions

Control functions for the EthercatDevice\_DmpCIQ\_Generic class.

Functions:

- [update\(\)](#)

## update()

### Description

Update state machines and internal variables for each function on the EtherCAT SubDevice.

### Syntax

```
int update();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

### 2.3.4.3 Digital I/O Functions

Digital I/O access functions for the EthercatDevice\_DmpCIQ\_Generic class.

Functions:

- [digitalWrite\(\)](#)
- [digitalRead\(\)](#)
- [digitalWriteAll\(\)](#)
- [digitalReadAll\(\)](#)

## digitalWrite()

### Description

Write a **HIGH** or a **LOW** value to a specified digital output pin on the EtherCAT SubDevice.

### Syntax

```
int digitalWrite(int pin, uint8_t value);
```

### Parameters

- **[in] pin**  
The digital output pin number of the EtherCAT SubDevice.
- **[in] value**  
Digital logic level, the value is **HIGH** or **LOW**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWrite(0, HIGH);
  delay(1000);
  device.digitalWrite(0, LOW);
  delay(1000);
}
```

## digitalRead()

### Description

Read the value from a specified digital input pin on the EtherCAT SubDevice.

### Syntax

```
int digitalRead(int pin);
```

### Parameters

- [in] pin

The digital input pin number of the EtherCAT SubDevice.

### Return Value

Return the digital logic level of the specified digital pin, the value is HIGH or LOW.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWrite(0, HIGH);
  delay(1000);
  Serial.print("DI0: ");
  Serial.println(device.digitalRead(0));

  device.digitalWrite(0, LOW);
  delay(1000);
  Serial.print("DI0: ");
  Serial.println(device.digitalRead(0));
}
```

## digitalWriteAll()

### Description

Write the digital output state of all pins on the EtherCAT SubDevice simultaneously.

### Syntax

```
int digitalWriteAll(uint32_t value);
```

### Parameters

- [in] value

This parameter is a 32-bit unsigned integer that specifies the value to be written to all digital output pins. Each bit in the value corresponds to a digital output pin of the EtherCAT SubDevice, with the following mapping:

- Bit 0 indicates digital output pin 0.
- Bit 1 indicates digital output pin 1.
- And so on, up to bit 31 which indicates digital output pin 31.

A value of 1 typically sets the corresponding pin to **HIGH**, while a value of 0 sets it to **LOW**.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWriteAll(0x55555555);
  delay(1000);
  device.digitalWriteAll(0xAAAAAAAA);
  delay(1000);
}
```

## digitalReadAll()

### Description

Read the digital input state of all pins on the EtherCAT SubDevice simultaneously.

### Syntax

```
uint32_t digitalReadAll();
```

### Parameters

None.

### Return Value

Return a 32-bit unsigned integer that represents the combined state of all digital input pins. Each bit in the returned value corresponds to a digital input pin of the EtherCAT SubDevice, with the following mapping:

- Bit 0 indicates digital input pin 0.
- Bit 1 indicates digital input pin 1.
- And so on, up to bit 31 which indicates digital input pin 31.

A value of 1 indicates that the corresponding pin is currently **HIGH**, while a value of 0 indicates that it is currently **LOW**.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.digitalWriteAll(0x55555555);
  delay(1000);
  Serial.print("DI: ");
  Serial.println(device.digitalReadAll(), HEX);
}
```

```
device.digitalWriteAll(0xAFFFFFFF);  
delay(1000);  
Serial.print("DI: ");  
Serial.println(device.digitalReadAll(), HEX);  
}
```

## 2.3.4.4 Analog I/O Functions

Analog I/O access functions for the EthercatDevice\_DmpCIQ\_Generic class.

Functions:

- [analogWrite\(\)](#)
- [voltageWrite\(\)](#)
- [currentWrite\(\)](#)
- [analogRead\(\)](#)
- [voltageRead\(\)](#)

## analogWrite()

### Description

Write a value to the specified analog output channel on the EtherCAT SubDevice.

### Syntax

```
int analogWrite(int ch, int32_t value);
```

### Parameters

- **[in] ch**  
The specified analog output channel on the EtherCAT SubDevice.
- **[in] value**  
The analog output value to be written. This parameter is a 32-bit signed integer whose value is linearly mapped to a physical output by the following rule, according to the mode configuration of the specified channel:
  - Voltage Mode: Maps a value of 0 to voltage 0V,  $2^{31}$  to 64V, and  $-2^{31}$  to -64V. (i.e.,  $2^{25}$  would get mapped to 1V.)
  - Current Mode: Maps a value of 0 to current 0A,  $2^{31}$  to 16A, and  $-2^{31}$  to -16A. (i.e.,  $2^{27}$  would get mapped to 1A.)

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

int voltage = 5;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.analogWrite(0, voltage << 25);
  delay(1000);
}
```

```
device.analogWrite(0, 0);  
delay(1000);  
  
device.analogWrite(0, -1 * (voltage << 25));  
delay(1000);  
  
device.analogWrite(0, 0);  
delay(1000);  
}
```

## voltageWrite()

### Description

Write a voltage value to the specified analog output channel on the EtherCAT SubDevice. This function is used to specify that the channel is configured for Voltage Mode.

### Syntax

```
int voltageWrite(int ch, double voltage);
```

### Parameters

- `[in] ch`  
The specified analog output channel on the EtherCAT SubDevice.
- `[in] voltage`  
The desired output voltage, in volts (V).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.voltageWrite(0, 10.0);
  delay(1000);

  device.voltageWrite(0, 0);
  delay(1000);

  device.voltageWrite(0, -10.0);
  delay(1000);
}
```

```
device.voltageWrite(0, 0);  
delay(1000);  
}
```

## currentWrite()

### Description

Write a current value to the specified analog output channel on the EtherCAT SubDevice. This function is used to specify that the channel is configured for Current Mode.

### Syntax

```
int currentWrite(int ch, double current);
```

### Parameters

- `[in] ch`  
The specified analog output channel on the EtherCAT SubDevice.
- `[in] current`  
The desired output current, in amperes (A).

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.currentWrite(0, 0.02);
  delay(1000);

  device.currentWrite(0, 0);
  delay(1000);

  device.currentWrite(0, 0.01);
  delay(1000);
}
```

```
device.currentWrite(0, 0);  
delay(1000);  
}
```

## analogRead()

### Description

Read the value of the specified analog input channel on the EtherCAT SubDevice.

### Syntax

```
int analogRead(int ch);
```

### Parameters

- `[in] ch`

The specified analog input channel on the EtherCAT SubDevice.

### Return Value

Return a 32-bit signed integer within the range of  $-2^{31}$  to  $2^{31}$  as the analog input value. This value is linearly mapped to a physical input voltage in volts, where 0 maps to 0V,  $2^{31}$  maps to 64V, and  $-2^{31}$  maps to -64V.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AI0: ");
  Serial.println(device.analogRead(0));
  delay(1000);
}
```

## voltageRead()

### Description

Read the voltage value of the specified analog input channel on the EtherCAT SubDevice.

### Syntax

```
double voltageRead(int ch);
```

### Parameters

- [in] ch

The specified analog input channel on the EtherCAT SubDevice.

### Return Value

Return the voltage value expressed in volts (V) as a double.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and cannot be called in FPU-disabled callback functions. For more details about FPU-disabled callback functions, please refer to [Callback Functions](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("AIO: ");
  Serial.println(device.voltageRead(0));
  delay(1000);
}
```

## 2.3.4.5 UART Functions

UART access functions for the EthercatDevice\_DmpCIQ\_Generic class.

Functions:

- [uartIs485\(\)](#)
- [uartSetBaud\(\)](#)
- [uartSetFormat\(\)](#)
- [uartSetFlowControl\(\)](#)
- [uartGetRTS\(\)](#)
- [uartGetCTS\(\)](#)
- [uartGetDTR\(\)](#)
- [uartGetDSR\(\)](#)
- [uartSetRTS\(\)](#)
- [uartSetDTR\(\)](#)
- [uartClearFIFO\(\)](#)
- [uartClearTxQueue\(\)](#)
- [uartClearRxQueue\(\)](#)
- [uartQueryTxQueue\(\)](#)
- [uartQueryRxQueue\(\)](#)
- [uartTxQueueEmpty\(\)](#)
- [uartRxQueueEmpty\(\)](#)
- [uartTxQueueFull\(\)](#)
- [uartRxQueueFull\(\)](#)
- [uartSend\(\)](#)
- [uartWrite\(\)](#)
- [uartReceive\(\)](#)
- [uartRead\(\)](#)

## uartIs485()

### Description

Check if the specified UART port on the EtherCAT SubDevice is in RS485 mode.

### Syntax

```
int uartIs485(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return whether the specified UART port of the EtherCAT SubDevice is in RS485 mode.

- 1 means RS485.
- 0 means non-RS485.

if the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("RS485 mode: ");
  Serial.println(device.uartIs485(0));
}

void loop() {
}
```

## uartSetBaud()

### Description

Configure the baud rate for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetBaud(int dev, uint32_t baud);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- `[in] baud`  
The baud rate to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.uartSetBaud(0, 115200);
}

void loop() {
}
```

## uartSetFormat()

### Description

Configure the UART frame format for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetFormat(int dev, uint8_t format);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- **[in] format**  
The UART frame format to be configured. The bit definition of this parameter is as follows:

7	6	5	4	3	2	1	0
Reserved	Parity			Stop	Data Bits		

**Data Bits:** These bits define the word length of the data being transmitted and received.

Definition	Value	Description
ECAT_UART_BYTESIZE5	0x00	5 data bits.
ECAT_UART_BYTESIZE6	0x01	6 data bits.
ECAT_UART_BYTESIZE7	0x02	7 data bits.
ECAT_UART_BYTESIZE8	0x03	8 data bits.

**Stop:** This bit selects the number of stop bits to be transmitted.

Definition	Value	Description
ECAT_UART_STOPBIT1	0x00	One stop bit.
ECAT_UART_STOPBIT2	0x04	Two stop bits (1.5 with 5-bit data).

**Parity:** These bits select the way in which parity control is performed.

Definition	Value	Description
ECAT_UART_NOPARITY	0x00	No parity bit.
ECAT_UART_ODDPARITY	0x08	Odd parity.
ECAT_UART_EVENPARITY	0x18	Even parity.
ECAT_UART_MARKPARITY	0x28	The parity bit exists and is always 1.
ECAT_UART_SPACEPARITY	0x38	The parity bit exists and is always 0.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

**Comment**

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

**Example Code**

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.uartSetFormat(0, ECAT_UART_BYTESIZE8 + ECAT_UART_NOPARITY +
  ECAT_UART_STOPBIT1);
}

void loop() {
}
```

## uartSetFlowControl()

### Description

Configure the flow control mode for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetFlowControl(int dev, int control);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- `[in] control`  
The flow control mode to be configured.
  - **ECAT\_UART\_NO\_CONTROL** : Disable flow control.
  - **ECAT\_UART\_RTS\_CTS** : RTS/CTS flow control is a hardware flow control scheme that is commonly used in RS232.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QEER11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.uartSetFlowControl(0, ECAT_UART_RTS_CTS);
}

void loop() {
}
```

## uartGetRTS()

### Description

Get the current state of the RTS control signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartGetRTS(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return the current state of the RTS control signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  device.uartSetRTS(0, 1);
  Serial.print("RTS: ");
  Serial.println(device.uartGetRTS(0));
}
```

```
delay(1000);

device.uartSetRTS(0, 0);
Serial.print("RTS: ");
Serial.println(device.uartGetRTS(0));
delay(500);

// ...
}
```

## uartGetCTS()

### Description

Get the current state of the CTS signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartGetCTS(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return the current state of the CTS signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("CTS: ");
  Serial.println(device.uartGetCTS(0));
  delay(1000);
}
```

## uartGetDTR()

### Description

Get the current state of the DTR control signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartGetDTR(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return the current state of the DTR control signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    device.uartSetDTR(0, 1);
    Serial.print("DTR: ");
    Serial.println(device.uartGetDTR(0));
}
```

```
delay(1000);

device.uartSetDTR(0, 0);
Serial.print("DTR: ");
Serial.println(device.uartGetDTR(0));
delay(500);

// ...
}
```

## uartGetDSR()

### Description

Get the current state of the DSR signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartGetDSR(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return the current state of the DSR signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("DSR: ");
    Serial.println(device.uartGetDSR(0));
    delay(1000);
}
```

## uartSetRTS()

### Description

Control the RTS signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetRTS(int dev, uint8_t value);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- **[in] value**  
The RTS signal value to be set.  
0: Indicates an inactive RTS signal.  
1 to 255: Indicates an active RTS signal

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  device.uartSetRTS(0, 1);
  delay(1000);
}
```

```
device.uartSetRTS(0, 0);  
delay(500);  
  
// ...  
}
```

## uartSetDTR()

### Description

Control the DTR signal for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetDTR(int dev, uint8_t value);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- **[in] value**  
The DTR signal value to be set.  
0: Indicates an inactive DTR signal.  
1 to 255: Indicates an active DTR signal.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  device.uartSetDTR(0, 1);
  delay(1000);
}
```

```
device.uartSetDTR(0, 0);  
delay(500);  
  
// ...  
}
```

## uartClearFIFO()

### Description

Clear the TX and RX FIFOs for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartClearFIFO(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearFIFO(0);
}

void loop() {
  // ...
}
```

## uartClearTxQueue()

### Description

Clear the software TX FIFO for the specified UART port on the EtherCAT SubDevice in this library.

### Syntax

```
int uartClearTxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearTxQueue(0);
}

void loop() {
  // ...
}
```

## uartClearRxQueue()

### Description

Clear the software RX FIFO for the specified UART port on the EtherCAT SubDevice in this library.

### Syntax

```
int uartClearRxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearRxQueue(0);
}

void loop() {
  // ...
}
```

## uartQueryTxQueue()

### Description

Get the current number of bytes in the software TX FIFO for the specified UART port on the EtherCAT SubDevice in this library.

### Syntax

```
int uartQueryTxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return the current number of bytes in the software TX FIFO for the specified UART port. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("TX FIFO: ");
  Serial.println(device.uartQueryTxQueue(0));
}

void loop() {
}
```

## uartQueryRxQueue()

### Description

Get the current number of bytes in the software RX FIFO for the specified UART port on the EtherCAT SubDevice in this library.

### Syntax

```
int uartQueryRxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

The current number of bytes in the software RX FIFO for the specified UART port. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("RX FIFO: ");
  Serial.println(device.uartQueryRxQueue(0));
}

void loop() {
}
```

## uartTxQueueEmpty()

### Description

Check if the software TX FIFO for the specified UART port on the EtherCAT SubDevice in this library is empty.

### Syntax

```
int uartTxQueueEmpty(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return whether the software TX FIFO in this library is empty.

If the FIFO on the COM port is empty, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("TX FIFO is Empty: ");
  Serial.println(device.uartTxQueueEmpty(0));
}

void loop() {
}
```

## uartRxQueueEmpty()

### Description

Check if the software RX FIFO for the specified UART port on the EtherCAT SubDevice in this library is empty.

### Syntax

```
int uartRxQueueEmpty(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return whether the software RX FIFO in this library is empty.

If the FIFO on the COM port is empty, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("RX FIFO is Empty: ");
  Serial.println(device.uartRxQueueEmpty(0));
}

void loop() {
}
```

## uartTxQueueFull()

### Description

Check if the software TX FIFO for the specified UART port of the EtherCAT SubDevice in this library is full.

### Syntax

```
int uartTxQueueFull(int dev);
```

### Parameters

- [in] dev  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return whether the software TX FIFO in this library is full.

If the FIFO on the COM port is full, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("TX FIFO is Full: ");
  Serial.println(device.uartTxQueueFull(0));
}

void loop() {
}
```

## uartRxQueueFull()

### Description

Check if the software RX FIFO for the specified UART port of the EtherCAT SubDevice in this library is full.

### Syntax

```
int uartRxQueueFull(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return whether the software RX FIFO in this library is full.

If the FIFO on the COM port is full, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("RX FIFO is Full: ");
  Serial.println(device.uartRxQueueFull(0));
}

void loop() {
}
```

## uartSend()

### Description

Transmit multiple bytes data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSend(int dev, void *buf, size_t size);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- `[in] buf`  
The data buffer to be transmitted.
- `[in] size`  
The size of the data buffer.

### Return Value

Return the number of bytes transmitted. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
    device.update();
}

void setup() {
    char buffer[] = {"Hello world!"};

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.uartSend(0, buffer, strlen(buffer));
}
```

```
}  
  
void loop() {  
  // ...  
}
```

## uartWrite()

### Description

Transmit one byte data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartWrite(int dev, uint8_t value);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- `[in] value`  
The one byte data to be transmitted.

### Return Value

Return the number of bytes transmitted. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of `EthercatMaster::begin()`. This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  device.uartWrite(0, 'H');
  device.uartWrite(0, 'e');
  device.uartWrite(0, 'l');
  device.uartWrite(0, 'l');
  device.uartWrite(0, 'o');
}
```

```
void loop() {  
  // ...  
}
```

## uartReceive()

### Description

Receive multiple bytes data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartReceive(int dev, void *buf, size_t size);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.
- `[in] buf`  
The data buffer to be received.
- `[in] size`  
The size of the data buffer.

### Return Value

Return the number of bytes received. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}
```

```
void loop() {  
  char buffer[256];  
  int rc;  
  
  rc = device.uartReceive(0, buffer, 256);  
  for (int i = 0; i < rc; i++) {  
    Serial.print(buffer[i]);  
  }  
  // ...  
}
```

## uartRead()

### Description

Read one byte data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartRead(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.

### Return Value

Return one byte data. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11CFFG device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    int ch = device.uartRead(0);
    if (ch >= 0) {
```

```
Serial.print((char)ch);  
}  
// ...  
}
```

## 2.3.5 EthercatDevice\_DmpHID\_Generic

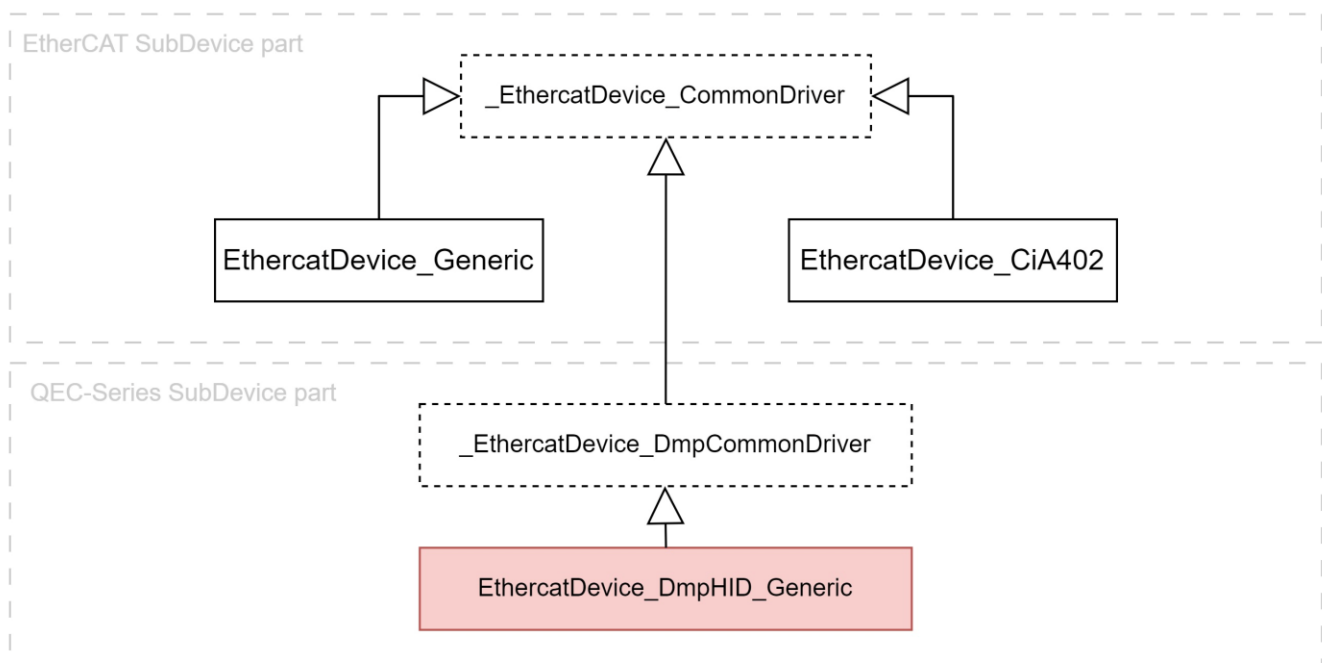
EthercatDevice\_DmpHID\_Generic is an EtherCAT SubDevice class specifically developed by ICOP for QEC EtherCAT SubDevice HID modules. It encompasses RS232/RS485, 4x4 Keypad, 16x2 LCM, Manual Pulse Generator (MPG), and Buzzer functionalities.

**RS232** and **RS485** are electrical specifications that define the voltage levels, signal timing, and connector pinouts for implementing UART communication over physical cables. **UART (Universal Asynchronous Receiver Transmitter)** is a hardware interface standard for asynchronous serial data communication. It defines the format of data bits, start and stop bits, parity bits, and baud rate for serial communication. UART is commonly used for connecting microcontrollers, computers, and other devices for data exchange. This device features two UART ports that can be freely switched between RS232 or RS485 modes to accommodate user application requirements. This device features a **4x4 keypad** that provides a user interface for inputting numbers, symbols, and characters. The keypad is arranged in a matrix format with four rows and four columns, with each key representing a specific character or function.

This device features a **2x16 LCD module** that provides a visual display for presenting information to the user. The LCM is arranged in a matrix format with two rows and 16 columns, allowing it to display up to 32 characters per line.

This device features a **Manual Pulse Generator (MPG)** interface that enables the connection of specific MPG devices, allowing precise control of machine or system movement. The interface that counts MPG pulses is referred to as an Encoder in this context. In addition to Encoder counter, the interface also offers Ratio knob, Axis knob, Emergency Stop switch, and Enable switch.

The class relationships of EthercatDevice\_DmpHID\_Generic are illustrated in the following diagram:



- EthercatDevice\_DmpHID\_Generic *inherits from* \_EthercatDevice\_DmpCommonDriver.

## Base Class:

- [\\_EthercatDevice\\_CommonDriver](#)

## Derived Class:

Class Name	Vendor ID	Product Code	UART	Keypad	LCM	MPG
EthercatDevice_QECR11HU1S	0x00000bc3	0x0086d404	0			0
EthercatDevice_QECR11HU5S	0x00000bc3	0x0086d403	0			
EthercatDevice_QECR00HU5S	0x00000bc3	0x0086d400	0			
EthercatDevice_QECR11HU9S	0x00000bc3	0x0086d402	0	0	0	0
EthercatDevice_QECR00HU9S	0x00000bc3	0x0086d401	0	0	0	0

## Function Groups:

- [Initialization](#)
- [Control](#)
- [UART](#)
- [LCM](#)
- [Keypad](#)
- [MPG](#)
- [Buzzer](#)

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">attach()</a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#">detach()</a>	Deinitialize the object of this EtherCAT SubDevice class.	
<b>Control-related functions</b>		
<a href="#">update()</a>	Update state machines and internal variables for each function.	0
<b>UART access functions</b>		
<a href="#">uartIs485()</a>	Check if the specified UART port is in RS485 mode.	0
<a href="#">uartSetBaud()</a>	Configure the baud rate.	
<a href="#">uartSetFormat()</a>	Configure the UART frame format.	
<a href="#">uartSetFlowControl()</a>	Configure the flow control mode.	
<a href="#">uartGetRTS()</a>	Get the current state of the RTS control signal.	0
<a href="#">uartGetCTS()</a>	Get the current state of the CTS signal.	0
<a href="#">uartGetDTR()</a>	Get the current state of the DTR control signal.	0
<a href="#">uartGetDSR()</a>	Get the current state of the DSR signal.	0
<a href="#">uartSetRTS()</a>	Control the RTS signal.	0
<a href="#">uartSetDTR()</a>	Control the DTR signal.	0
<a href="#">uartClearFIFO()</a>	Clear the TX and RX FIFOs.	
<a href="#">uartClearTxQueue()</a>	Clear the software TX FIFO.	0
<a href="#">uartClearRxQueue()</a>	Clear the software RX FIFO.	0
<a href="#">uartQueryTxQueue()</a>	Get the current number of bytes in the software TX FIFO.	0
<a href="#">uartQueryRxQueue()</a>	Get the current number of bytes in the software RX FIFO.	0
<a href="#">uartTxQueueEmpty()</a>	Check if the software TX FIFO is empty.	0
<a href="#">uartRxQueueEmpty()</a>	Check if the software RX FIFO is empty.	0
<a href="#">uartTxQueueFull()</a>	Check if the software TX FIFO is full.	0
<a href="#">uartRxQueueFull()</a>	Check if the software RX FIFO is full.	0
<a href="#">uartSend()</a>	Transmit multiple byte data.	0
<a href="#">uartWrite()</a>	Transmit one byte data.	0
<a href="#">uartReceive()</a>	Receive multiple byte data.	0
<a href="#">uartRead()</a>	Read one byte data.	0
<b>LCM access functions</b>		
<a href="#">lcmHeight()</a>	Get the height of the LCM.	0
<a href="#">lcmWidth()</a>	Get the width of the LCM.	0
<a href="#">lcmWordWrap()</a>	Enable or disable the word wrap feature.	0
<a href="#">lcmGotoXY()</a>	Move the current cursor to the specified coordinates.	0
<a href="#">lcmClear()</a>	Clear the screen.	0
<a href="#">lcmPrint()</a>	Print a string.	0
<a href="#">lcmWrite()</a>	Print a character.	0
<b>Keypad access functions</b>		

<a href="#">keypadSetTimeout()</a>	Set the timeout for input data buffer.	0
<a href="#">keypadClear()</a>	Clear the input data buffer.	0
<a href="#">keypadRead()</a>	Read one byte input data.	0
<b>MPG access functions</b>		
<a href="#">mpgSetCallback()</a>	Register the MPG event callback function.	0
<a href="#">mpgSetNoiseFilter()</a>	Configure the noise filter.	
<a href="#">mpgInvertEncoderDirection()</a>	Invert the encoder's counting direction.	
<a href="#">mpgWriteEncoderRaw()</a>	Write the encoder's raw data.	
<a href="#">mpgWriteEncoder()</a>	Write the logical counter value of the encoder.	0
<a href="#">mpgReadEncoderRaw()</a>	Read the encoder's raw data.	0
<a href="#">mpgReadEncoder()</a>	Read the logical counter value of the encoder.	0
<a href="#">mpgReadEmergencyStop()</a>	Read the emergency stop status.	0
<a href="#">mpgReadEnableSwitch()</a>	Read the enable switch status.	0
<a href="#">mpgReadAxis()</a>	Read the axis value.	0
<a href="#">mpgReadRatio()</a>	Read the ratio value.	0
<b>Buzzer access functions</b>		
<a href="#">buzzer()</a>	Emit a sound of the specified frequency.	0

## 2.3.5.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)

## attach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::attach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR00HU9S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00HU9S device;

void setup() {
    master.begin();
    device.attach(0, master);
    master.start();
}

void loop() {
    // ...
}
```

## detach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::detach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR00HU9S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR00HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);

  delay(3000);

  device.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 2.3.5.2 Control Functions

Control functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [update\(\)](#)

## update()

### Description

Update state machines and internal variables for each function on the EtherCAT SubDevice.

### Syntax

```
int update();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

## 2.3.5.3 UART Functions

UART access functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [uartIs485\(\)](#)
- [uartSetBaud\(\)](#)
- [uartSetFormat\(\)](#)
- [uartSetFlowControl\(\)](#)
- [uartGetRTS\(\)](#)
- [uartGetCTS\(\)](#)
- [uartGetDTR\(\)](#)
- [uartGetDSR\(\)](#)
- [uartSetRTS\(\)](#)
- [uartSetDTR\(\)](#)
- [uartClearFIFO\(\)](#)
- [uartClearTxQueue\(\)](#)
- [uartClearRxQueue\(\)](#)
- [uartQueryTxQueue\(\)](#)
- [uartQueryRxQueue\(\)](#)
- [uartTxQueueEmpty\(\)](#)
- [uartRxQueueEmpty\(\)](#)
- [uartTxQueueFull\(\)](#)
- [uartRxQueueFull\(\)](#)
- [uartSend\(\)](#)
- [uartWrite\(\)](#)
- [uartReceive\(\)](#)
- [uartRead\(\)](#)

## uartIs485()

### Description

Check if the specified UART port of the EtherCAT SubDevice is in RS485 mode.

### Syntax

```
int uartIs485(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return whether the specified UART port of the EtherCAT SubDevice is in RS485 mode.

- 1 means RS485.
- 0 means non-RS485.

if the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("RS485 mode: ");
  Serial.println(device.uartIs485(0));
}

void loop() {
  // ...
}
```

## uartSetBaud()

### Description

Configure the baud rate for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSetBaud(int dev, uint32_t baud);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- `[in] baud`  
The baud rate to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartSetBaud(0, 115200);
}

void loop() {
  // ...
}
```

## uartSetFormat()

### Description

Configure the UART frame format for the specified UART port on the EtherCAT SubDevice.

### Syntax

```
int uartSetFormat(int dev, uint8_t format);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- **[in] format**  
The UART frame format to be configured. The bit definition of this parameter is as follows:

7	6	5	4	3	2	1	0
Reserved		Parity			Stop	Data Bits	

**Data Bits:** These bits define the word length of the data being transmitted and received.

Definition	Value	Description
ECAT_UART_BYTESIZE5	0x00	5 data bits.
ECAT_UART_BYTESIZE6	0x01	6 data bits.
ECAT_UART_BYTESIZE7	0x02	7 data bits.
ECAT_UART_BYTESIZE8	0x03	8 data bits.

**Stop:** This bit selects the number of stop bits to be transmitted.

Definition	Value	Description
ECAT_UART_STOPBIT1	0x00	One stop bit.
ECAT_UART_STOPBIT2	0x04	Two stop bits (1.5 with 5-bit data).

**Parity:** These bits select the way in which parity control is performed.

Definition	Value	Description
ECAT_UART_NOPARITY	0x00	No parity bit.
ECAT_UART_ODDPARITY	0x08	Odd parity.
ECAT_UART_EVENPARITY	0x18	Even parity.
ECAT_UART_MARKPARITY	0x28	The parity bit exists and is always 1.
ECAT_UART_SPACEPARITY	0x38	The parity bit exists and is always 0.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

## Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartSetFormat(0, ECAT_UART_BYTESIZE8 + ECAT_UART_NOPARITY +
  ECAT_UART_STOPBIT1);
}

void loop() {
  // ...
}
```

## uartSetFlowControl()

### Description

Configure the flow control mode for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSetFlowControl(int dev, int control);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- **[in] control**  
The flow control mode to be configured.
  - **ECAT\_UART\_NO\_CONTROL** : Disable flow control.
  - **ECAT\_UART\_RTS\_CTS** : RTS/CTS flow control is a hardware flow control scheme that is commonly used in RS232.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartSetFlowControl(0, ECAT_UART_RTS_CTS);
}

void loop() {
  // ...
}
```

## uartGetRTS()

### Description

Get the current state of the RTS control signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartGetRTS(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return the current state of the RTS control signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  device.uartSetRTS(0, 1);
  Serial.print("RTS: ");
```

```
Serial.println(device.uartGetRTS(0));  
delay(1000);  
  
device.uartSetRTS(0, 0);  
Serial.print("RTS: ");  
Serial.println(device.uartGetRTS(0));  
delay(500);  
// ...  
}
```

## uartGetCTS()

### Description

Get the current state of the CTS signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartGetCTS(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return the current state of the CTS signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("CTS: ");
    Serial.println(device.uartGetCTS(0));
    delay(1000);
}
```

```
// ...  
}
```

## uartGetDTR()

### Description

Get the current state of the DTR control signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartGetDTR(int dev);
```

### Parameters

- [in] dev  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return the current state of the DTR control signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    device.uartSetDTR(0, 1);
    Serial.print("DTR: ");
```

```
Serial.println(device.uartGetDTR(0));  
delay(1000);  
  
device.uartSetDTR(0, 0);  
Serial.print("DTR: ");  
Serial.println(device.uartGetDTR(0));  
delay(500);  
// ...  
}
```

## uartGetDSR()

### Description

Get the current state of the DSR signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartGetDSR(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return the current state of the DSR signal. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  Serial.print("DSR: ");
  Serial.println(device.uartGetDSR(0));
  delay(1000);
}
```

```
// ...  
}
```

## uartSetRTS()

### Description

Control the RTS signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSetRTS(int dev, uint8_t value);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- **[in] value**  
The RTS signal value to be set.  
0: Indicates an inactive RTS signal.  
1 to 255: Indicates an active RTS signal

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  device.uartSetRTS(0, 1);
}
```

```
delay(1000);  
device.uartSetRTS(0, 0);  
delay(500);  
// ...  
}
```

## uartSetDTR()

### Description

Control the DTR signal for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSetDTR(int dev, uint8_t value);
```

### Parameters

- **[in] dev**  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- **[in] value**  
The DTR signal value to be set.  
0: Indicates an inactive DTR signal.  
1 to 255: Indicates an active DTR signal.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    device.uartSetDTR(0, 1);
}
```

```
delay(1000);  
device.uartSetDTR(0, 0);  
delay(500);  
// ...  
}
```

## uartClearFIFO()

### Description

Clear the TX and RX FIFOs for the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartClearFIFO(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearFIFO(0);
}

void loop() {
  // ...
}
```

## uartClearTxQueue()

### Description

Clear the software TX FIFO for the specified UART port of the EtherCAT SubDevice in this library.

### Syntax

```
int uartClearTxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearTxQueue(0);
}

void loop() {
  // ...
}
```

## uartClearRxQueue()

### Description

Clear the software RX FIFO for the specified UART port of the EtherCAT SubDevice in this library.

### Syntax

```
int uartClearRxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.uartClearRxQueue(0);
}

void loop() {
  // ...
}
```

## uartQueryTxQueue()

### Description

Get the current number of bytes in the software TX FIFO for the specified UART port of the EtherCAT SubDevice in this library.

### Syntax

```
int uartQueryTxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return the current number of bytes in the software TX FIFO for the specified UART port. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("TX FIFO: ");
  Serial.println(device.uartQueryTxQueue(0));
}

void loop() {
  // ...
}
```

## uartQueryRxQueue()

### Description

Get the current number of bytes in the software RX FIFO for the specified UART port of the EtherCAT SubDevice in this library.

### Syntax

```
int uartQueryRxQueue(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

The current number of bytes in the software RX FIFO for the specified UART port. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("RX FIFO: ");
  Serial.println(device.uartQueryRxQueue(0));
}

void loop() {
  // ...
}
```

## uartTxQueueEmpty()

### Description

Check if the software TX FIFO for the specified UART port of the EtherCAT SubDevice in this library is empty.

### Syntax

```
int uartTxQueueEmpty(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return whether the software TX FIFO in this library is empty.

If the FIFO on the COM port is empty, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("TX FIFO is Empty: ");
  Serial.println(device.uartTxQueueEmpty(0));
}

void loop() {
  // ...
}
```

## uartRxQueueEmpty()

### Description

Check if the software RX FIFO for the specified UART port on the EtherCAT SubDevice in this library is empty.

### Syntax

```
int uartRxQueueEmpty(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return whether the software RX FIFO in this library is empty.

If the FIFO on the COM port is empty, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("RX FIFO is Empty: ");
  Serial.println(device.uartRxQueueEmpty(0));
}

void loop() {
  // ...
}
```

## uartTxQueueFull()

### Description

Check if the software TX FIFO for the specified UART port of the EtherCAT SubDevice in this library is full.

### Syntax

```
int uartTxQueueFull(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return whether the software TX FIFO in this library is full.

If the FIFO on the COM port is full, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("TX FIFO is Full: ");
  Serial.println(device.uartTxQueueFull(0));
}
void loop() {
  // ...
}
```

## uartRxQueueFull()

### Description

Check if the software RX FIFO for the specified UART port of the EtherCAT SubDevice in this library is full.

### Syntax

```
int uartRxQueueFull(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return whether the software RX FIFO in this library is full.

If the FIFO on the COM port is full, the returned value is 1; if it is not, it is 0.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("RX FIFO is Full: ");
  Serial.println(device.uartRxQueueFull(0));
}

void loop() {
  // ...
}
```

## uartSend()

### Description

Transmit multiple bytes data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartSend(int dev, void *buf, size_t size);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- `[in] buf`  
The data buffer to be transmitted.
- `[in] size`  
The size of the data buffer.

### Return Value

Return the number of bytes transmitted. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    char buffer[] = {"Hello world!"};

    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
```

```
master.start();

device.uartSend(0, buffer, strlen(buffer));

Serial.print("Sent: ");
Serial.println(buffer);
}

void loop() {
  // ...
}
```

## uartWrite()

### Description

Transmit one byte data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartWrite(int dev, uint8_t value);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- `[in] value`  
The one byte data to be transmitted.

### Return Value

Return the number of bytes transmitted. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  device.uartWrite(0, 'H');
  device.uartWrite(0, 'e');
```

```
device.uartWrite(0, 'l');  
device.uartWrite(0, 'l');  
device.uartWrite(0, 'o');  
}  
  
void loop() {  
  // ...  
}
```

## uartReceive()

### Description

Receive multiple bytes data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartReceive(int dev, void *buf, size_t size);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.
- `[in] buf`  
The data buffer to be received.
- `[in] size`  
The size of the data buffer.

### Return Value

Return the number of bytes received. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

char buffer[256];
int rc;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
```

```
    master.start();
}

void loop() {
    rc = device.uartReceive(0, buffer, 256);
    for (int i = 0; i < rc; i++) {
        Serial.print(buffer[i]);
    }

    // ...
}
```

## uartRead()

### Description

Read one byte data from the specified UART port of the EtherCAT SubDevice.

### Syntax

```
int uartRead(int dev);
```

### Parameters

- `[in] dev`  
The specified UART port of the EtherCAT SubDevice.  
0 for COM1.  
1 for COM2.

### Return Value

Return one byte data. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    int ch = device.uartRead(0);
    if (ch >= 0)
        Serial.print((char)ch);
    // ...
}
```

## 2.3.5.4 LCM Functions

LCM access functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [lcmHeight\(\)](#)
- [lcmWidth\(\)](#)
- [lcmWordWrap\(\)](#)
- [lcmGotoXY\(\)](#)
- [lcmClear\(\)](#)
- [lcmPrint\(\)](#)
- [lcmWrite\(\)](#)

## lcmHeight()

### Description

Get the height of the LCM on the EtherCAT SubDevice.

### Syntax

```
int lcmHeight();
```

### Parameters

None.

### Return Value

Return the height of the LCM. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QEER11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("LCM Height: ");
  Serial.println(device.lcmHeight());
}

void loop() {
  // ...
}
```

## lcmWidth()

### Description

Get the width of the LCM on the EtherCAT SubDevice.

### Syntax

```
int lcmWidth();
```

### Parameters

None.

### Return Value

Return the width of the LCM. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  Serial.print("LCM Width: ");
  Serial.println(device.lcmWidth());
}

void loop() {
  // ...
}
```

## lcmWordWrap()

### Description

Enable or disable the word wrap feature on the LCM of the EtherCAT SubDevice.

### Syntax

```
int lcmWordWrap(bool wrap);
```

### Parameters

- `[in] wrap`

A Boolean value that specifies whether to enable or disable the word wrap feature on the LCM.

- `true`: The word wrap feature will be enabled.
- `false`: The word wrap feature will be disabled.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    device.lcmWordWrap(true);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.lcmPrint("Hi, this is QEC HID device.");
}

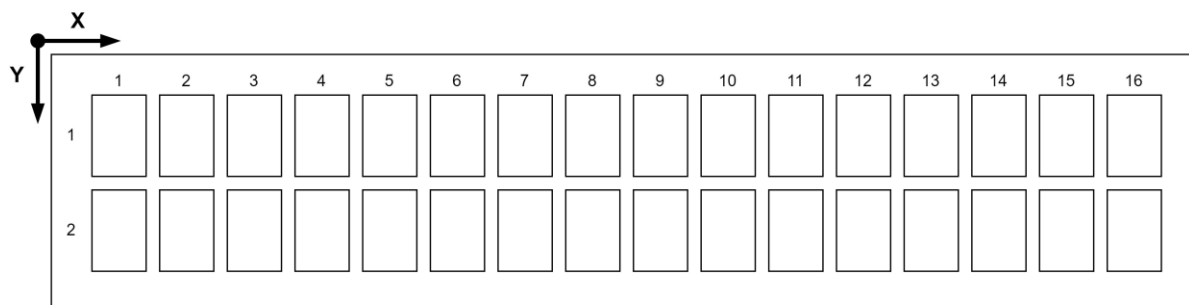
void loop() {
    // ...
}
```

```
}
```

## lcmGotoXY()

### Description

Move the current cursor on the LCM of the EtherCAT SubDevice to the specified coordinates.



### Syntax

```
int lcmGotoXY(int x, int y);
```

### Parameters

- `[in] x`  
X-axis position of the LCM.
- `[in] y`  
Y-axis position of the LCM.

### Return Value

Return the current cursor position of the LCM. The formula for calculating the cursor position

$P_{cursor}$  is as follows, where  $W$  is the width of the LCM:

$$P_{cursor} = W \times (Y - 1) + (X - 1)$$

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
```

```
master.begin();
device.attach(0, master);
master.attachCyclicCallback(CyclicCallback);
master.start();

device.lcmGotoXY(1, 1);
device.lcmPrint("Hello");
device.lcmGotoXY(8, 1);
device.lcmPrint("World!");
device.lcmGotoXY(2, 2);
device.lcmPrint("QEC HID device");
}

void loop() {
  // ...
}
```

## lcmClear()

### Description

Clear the screen of the LCM on the EtherCAT SubDevice.

### Syntax

```
int lcmClear();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.lcmClear();
}

void loop() {
    // ...
}
```

## lcmPrint()

### Description

Print the specified string to the LCM screen of the EtherCAT SubDevice.

### Syntax

```
int lcmPrint(const char *fmt, ...);
```

### Parameters

- `[in] fmt`  
The string to be printed on the LCM screen. This is a pointer to a null-terminated string containing the format specification for the output. The format string follows the same format as the `printf` function in C, allowing for insertion of variables and formatting options.
- `[in] ...`  
This is a variable number of arguments that will be inserted into the formatted string according to the format specifiers in the `fmt` string.

### Return Value

Return the number of characters printed to the LCM screen. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of `EthercatMaster::begin()`. This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.lcmPrint("Hello world!");
}
```

```
void loop() {  
  // ...  
}
```

## lcmWrite()

### Description

Print one character to the LCM screen of the EtherCAT SubDevice.

### Syntax

```
int lcmWrite(char c);
```

### Parameters

- `[in] c`

The character to be printed on the LCM screen.

### Return Value

Return the number of characters printed to the LCM screen. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.lcmWrite('H');
    device.lcmWrite('e');
    device.lcmWrite('l');
    device.lcmWrite('l');
    device.lcmWrite('o');
    device.lcmWrite('!');
}

void loop() {
```

```
// ...  
}
```

## 2.3.5.5 Keypad Functions

Keypad access functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [keypadSetTimeout\(\)](#)
- [keypadClear\(\)](#)
- [keypadRead\(\)](#)

## keypadSetTimeout()

### Description

Set the keypad input data buffer timeout for the EtherCAT SubDevice. If no keypad input data is read for a specified period, the input data buffer will be automatically cleared. The default timeout is 1000 milliseconds.

### Syntax

```
int keypadSetTimeout(uint32_t timeout_ms);
```

### Parameters

- [in] `timeout_ms`

Timeout in milliseconds. If this parameter is 0, it indicates that the keypad input data buffer will not be automatically cleared.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.keypadSetTimeout(3000);
}

void loop() {
  // ...
}
```

## keypadClear()

### Description

Clear the input data buffer of the keypad of the EtherCAT SubDevice.

### Syntax

```
int keypadClear();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.keypadClear();
}

void loop() {
  // ...
}
```

## keypadRead()

### Description

Read the input character from the keypad of the EtherCAT SubDevice.

### Syntax

```
int keypadRead();
```

### Parameters

None.

### Return Value

Read a character from the keypad. Return '\0' if no data available. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    int ch;
    if ((ch = device.keypadRead()) != '\0') {
        Serial.print((char)ch);
    }
}
```

```
}  
}
```

## 2.3.5.6 MPG Functions

MPG access functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [mpgSetCallback\(\)](#)
- [mpgSetNoiseFilter\(\)](#)
- [mpgInvertEncoderDirection\(\)](#)
- [mpgWriteEncoderRaw\(\)](#)
- [mpgWriteEncoder\(\)](#)
- [mpgReadEncoderRaw\(\)](#)
- [mpgReadEncoder\(\)](#)
- [mpgReadEmergencyStop\(\)](#)
- [mpgReadEnableSwitch\(\)](#)
- [mpgReadAxis\(\)](#)
- [mpgReadRatio\(\)](#)

## mpgSetCallback()

### Description

Register the MPG event callback function for the EtherCAT SubDevice.

### Syntax

```
int mpgSetCallback(void (*callback)(int));
```

### Parameters

- `[in] callback`

The MPG event callback function to be registered has an integer-type parameter that indicates the event type. The supported MPG event types are as follows:

Definition	Code	Description
ECAT_MPG_AXIS_CHANGE	1	The status of the Axis knob has changed.
ECAT_MPG_RATIO_CHANGE	2	The status of the Ratio knob has changed.
ECAT_MPG_EMERGENCY_STOP_CHANGE	3	The status of the Emergency Stop switch has changed.
ECAT_MPG_ENABLE_SWITCH_CHANGE	4	The status of the Enable switch has changed.
ECAT_MPG_ENCODER_CHANGE	5	The logical counter of the Encoder has changed.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

Since this callback function is invoked within the [update\(\)](#) function, if the [update\(\)](#) function is called in an interrupt callback function, specific usage restrictions must be adhered to. Please refer to [Callback Functions](#) for details.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void MpgCallback(int event) {
  switch (event) {
    case ECAT_MPG_AXIS_CHANGE:
      Serial.print("MPG Axis changed. ");
```

```
    Serial.println(device.mpgReadAxis());
    break;
case ECAT_MPG_RATIO_CHANGE:
    Serial.print("MPG Ratio changed. ");
    Serial.println(device.mpgReadRatio());
    break;
case ECAT_MPG_EMERGENCY_STOP_CHANGE:
    Serial.print("MPG Emergency Stop changed ");
    Serial.println(device.mpgReadEmergencyStop());
    break;
case ECAT_MPG_ENABLE_SWITCH_CHANGE:
    Serial.print("MPG Enable Switch changed. ");
    Serial.println(device.mpgReadEnableSwitch());
    break;
case ECAT_MPG_ENCODER_CHANGE:
    Serial.print("MPG Encoder changed. ");
    Serial.println(device.mpgReadEncoder());
    break;
}
}
void setup() {
    Serial.begin(115200);
    master.begin();
    device.attach(0, master);
    device.mpgSetCallback(MpgCallback);
    master.start();
}

void loop() {
    device.update();
}
```

## mpgSetNoiseFilter()

### Description

Configure the noise filter of the MPG on the EtherCAT SubDevice, including the noise filtering for the Emergency Stop switch, Axis knob, and Ratio knob.

### Syntax

```
int mpgSetNoiseFilter(uint32_t time_us);
```

### Parameters

- [in] `time_us`

The desired configuration time for the noise filter.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.mpgSetNoiseFilter(10000);
  master.start();
}

void loop() {
  device.update();
}
```

## mpgInvertEncoderDirection()

### Description

Invert the encoder's counting direction of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgInvertEncoderDirection(bool invert);
```

### Parameters

- `[in] invert`

A Boolean value that specifies whether to invert the counting direction of the Encoder.

- `true`: The counting direction will be inverted.
- `false`: The counting direction will not be changed.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.mpgInvertEncoderDirection(true);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

## mpgWriteEncoderRaw()

### Description

Write the encoder's raw data of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgWriteEncoderRaw(int32_t value);
```

### Parameters

- `[in] value`

The encoder's raw data to be written.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.mpgWriteEncoderRaw(100);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

## mpgWriteEncoder()

### Description

Write the logical counter to the Encoder of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgWriteEncoder(int32_t value);
```

### Parameters

- `[in] value`

The logical counter value of the encoder to be written.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.mpgWriteEncoder(100);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

## mpgReadEncoderRaw()

### Description

Read the encoder's raw data of the MPG on the EtherCAT SubDevice.

### Syntax

```
int32_t mpgReadEncoderRaw();
```

### Parameters

None.

### Return Value

Return the encoder's raw data.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

int32_t raw = 0;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    int32_t newRaw = device.mpgReadEncoderRaw();
    if (raw != newRaw) {
        raw = newRaw;
        Serial.print("Raw: ");
    }
}
```

```
Serial.println(raw);  
}  
// ...  
}
```

## mpgReadEncoder()

### Description

Read the logical counter value of the encoder of the MPG on the EtherCAT SubDevice.

### Syntax

```
int32_t mpgReadEncoder();
```

### Parameters

None.

### Return Value

Return the logical counter value of the encoder with 32-bit signed integer.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

int32_t encoder = 0;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    int32_t newEncoder = device.mpgReadEncoder();
    if (encoder != newEncoder) {
        encoder = newEncoder;
        Serial.print("Encoder: ");
    }
}
```

```
Serial.println(encoder);  
}  
  
// ...  
}
```

## mpgReadEmergencyStop()

### Description

Read the emergency stop status of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgReadEmergencyStop();
```

### Parameters

None.

### Return Value

Return the emergency stop status. 1 means the emergency stop signal pulls HIGH, and 0 means LOW. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("Emergency Stop: ");
    Serial.println(device.mpgReadEmergencyStop());
    delay(1000);
    // ...
}
```

## mpgReadEnableSwitch()

### Description

Read the enable switch status of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgReadEnableSwitch();
```

### Parameters

None.

### Return Value

Return the enable switch status. 1 means the enable switch signal pulls **HIGH**, and 0 means **LOW**. If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("Enable: ");
    Serial.println(device.mpgReadEnableSwitch());
    delay(1000);
    // ...
}
```

## mpgReadAxis()

### Description

Read the axis value of the MPG on the EtherCAT SubDevice.

### Syntax

```
int mpgReadAxis();
```

### Parameters

None.

### Return Value

Return the axis value.

- 0: off.
- 1: X-axis.
- 2: Y-axis.
- 3: Z-axis.
- 4: 4-axis.
- 5: 5-axis.
- 6: 6-axis.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}
```

```
}  
  
void loop() {  
  Serial.print("Axis: ");  
  Serial.println(device.mpgReadAxis());  
  delay(1000);  
  // ...  
}
```

## mpgReadRatio()

### Description

Read the ratio value of the MPG for the EtherCAT SubDevice.

### Syntax

```
int mpgReadRatio();
```

### Parameters

None.

### Returns

Return the ratio value.

- 0: 1x.
- 1: 10x.
- 2: 100x.

If the return value is smaller than 0, it means an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    Serial.print("Ratio: ");
```

```
Serial.println(device.mpgReadRatio());  
delay(1000);  
// ...  
}
```

## 2.3.5.7 Buzzer Functions

Buzzer access functions for the EthercatDevice\_DmpHID\_Generic class.

Functions:

- [buzzer\(\)](#)

## buzzer()

### Description

Emit a sound of the specified frequency from the buzzer on the EtherCAT SubDevice.

### Syntax

```
int buzzer(uint32_t hz, uint32_t duration_ms = 0);
```

### Parameters

- `[in] hz`  
The frequency of the sound in Hz. If this parameter is set to 0, it indicates that the sound should be stopped. If the value of the frequency is greater than 100,000 (100K), this library would replace the input value with 100K.
- `[in] duration_ms`  
The duration of the sound in milliseconds. If this parameter is omitted or set to 0, the sound will continue indefinitely.  
The default is 0 ms.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11HU9S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

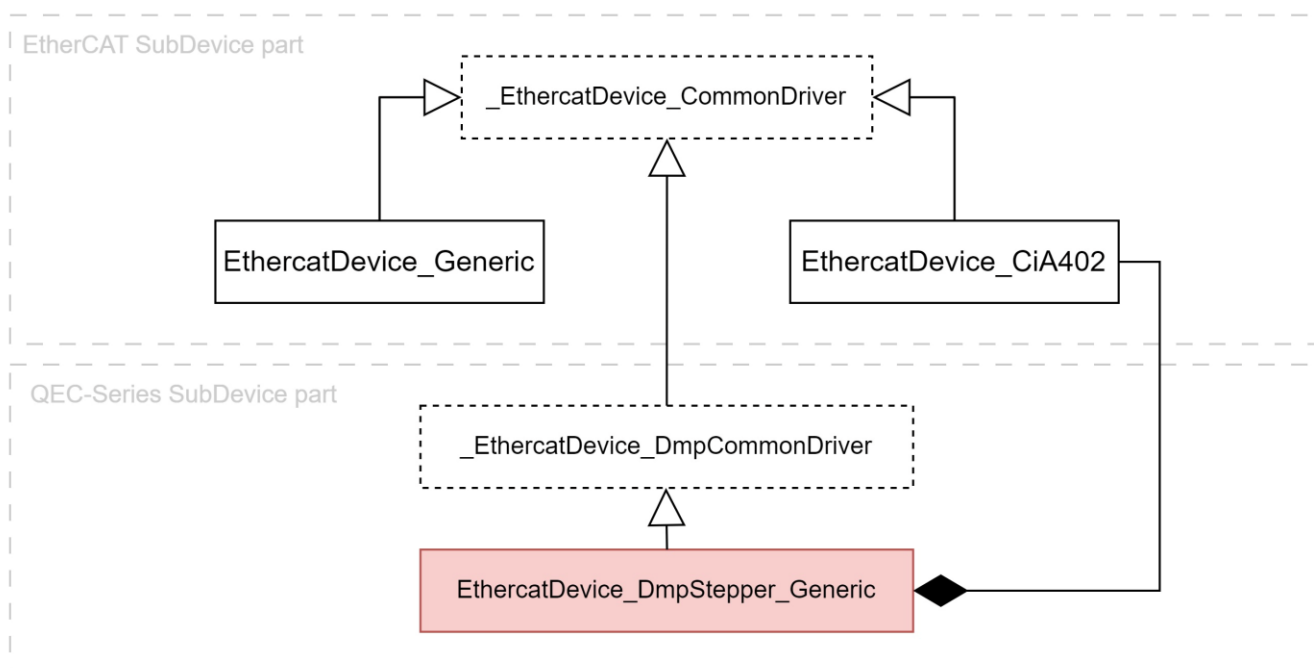
    device.buzzer(3000);
    delay(1000);
}
```

```
device.buzzer(500);  
delay(1000);  
device.buzzer(0);  
delay(1000);  
}  
  
void loop() {  
  // ...  
}
```

## 2.3.6 EthercatDevice\_DmpStepper\_Generic

EthercatDevice\_DmpStepper\_Generic is an EtherCAT SubDevice class developed by ICOP for 3-axis stepper motor controller EtherCAT SubDevice modules. This module features motor drivers, encoder inputs, and other CNC-related functions. In the motor control section, it not only supports the CiA 402 Servo mode but also the 3-axis synchronous G-code Controller mode.

The class relationships of EthercatDevice\_DmpStepper\_Generic are illustrated in the following diagram:



- EthercatDevice\_DmpStepper\_Generic inherits from \_EthercatDevice\_DmpCommonDriver.
- EthercatDevice\_DmpStepper\_Generic is composed of EthercatDevice\_CiA402.

Base Class:

- [\\_EthercatDevice\\_CommonDriver](#)

Derived Class:

Class Name	Vendor ID	Product Code
EthercatDevice_QECR11MP3S	0x00000bc3	0x0086d0d6
EthercatDevice_QECR00MP3S	0x00000bc3	0x0086d0d9
EthercatDevice_QECR11MV3S	0x00000bc3	0x0086d0e4
EthercatDevice_QECR00MV3S	0x00000bc3	0x0086d0e5

## Function Groups:

- [Initialization](#)
- [Control](#)
- [CiA 402](#)
- [Machine](#)
- [Encoder](#)
- [Configuration](#)

## Functions:

Function Name	Description	Callback Available
<b>Initialization-related functions</b>		
<a href="#">attach()</a>	Initialize the object of this EtherCAT SubDevice class.	
<a href="#">detach()</a>	Deinitialize the object of this EtherCAT SubDevice class.	
<a href="#">cia402GetServo()</a>	Obtain the CiA402 object pointer of the specified motor.	0 <sup>1</sup>
<b>Control-related functions</b>		
<a href="#">update()</a>	Update state machines and internal variables for each function.	0
<a href="#">attachInterrupt()</a>	Register the event callback function.	0
<b>G-code control functions</b>		
<a href="#">machineEnableSoftLimit()</a>	Enable the software limit function.	
<a href="#">machineDisableSoftLimit()</a>	Disable the software limit function.	
<a href="#">machineIsEmergencyStopped()</a>	Check if the machine is emergency stopped.	0
<a href="#">machineSetEmergencyStop()</a>	Initiate the emergency stop for the machine.	0
<a href="#">machineClearEmergencyStop()</a>	Clear the emergency stop for the machine.	0
<a href="#">machineIsLimitTouched()</a>	Check if the limit switch is touched.	0
<a href="#">machineIsHomingAttained()</a>	Check if the machine is homing attained.	0
<a href="#">machineIsServoOn()</a>	Check if the machine is servo-on.	0
<a href="#">machineIsMoving()</a>	Check if the machine is moving.	0
<a href="#">machineIsPositionErrorExceeded()</a>	Check if the position error has exceeded.	0
<a href="#">machineServoOn()</a>	Turn on all motors of the machine.	0
<a href="#">machineServoOff()</a>	Turn off all motors of the machine.	0
<a href="#">machineSetHomingSpeed()</a>	Set the homing speed.	
<a href="#">machineStartHoming()</a>	Initiate the homing procedure.	0
<a href="#">machineGcode()</a>	Send a G-code command.	0
<a href="#">machineActualPosition()</a>	Get the current position.	0
<b>Encoder access functions</b>		
<a href="#">encoderWrite()</a>	Write the encoder counter.	
<a href="#">encoderDirectionRead()</a>	Get the current encoder direction.	0

<a href="#">encoderRead()</a>	Read the encoder counter.	0
<b>Configuration-related functions</b>		
<a href="#">getDeviceMode()</a>	Get the device mode.	
<a href="#">configDeviceMode()</a>	Configure the device mode.	
<a href="#">configCiA402MotorResolution()</a>	Configure the motor resolution in CiA 402 Servo mode.	
<a href="#">configCiA402MotorPositionFeedbackSource()</a>	Configure the position feedback source for the specified motor.	
<a href="#">configCiA402MotorPositionFeedbackScale()</a>	Configure the position feedback scale of the specified motor.	
<a href="#">configCiA402MotorPositionFeedbackOffset()</a>	Configure the position feedback offset of the specified motor.	
<a href="#">configCiA402MotorSelfStartingSpeed()</a>	Configure the Self-Starting Speed for the specified motor.	
<a href="#">configMachineAxisMapping()</a>	Configure the mapping between mechanical axes and motors.	
<a href="#">configMachineDefaultFeedrate()</a>	Configure the default feed rate.	
<a href="#">configMachineDefaultHomingSpeed()</a>	Configure the default homing speed.	
<a href="#">configMachineHomingDirection()</a>	Configure the homing direction.	
<a href="#">configMachineHomingPriority()</a>	Configure the homing priority.	
<a href="#">configMachineMaxVelocity()</a>	Configure the maximum velocity.	
<a href="#">configMachineMaxAcceleration()</a>	Configure the maximum acceleration.	
<a href="#">configMachineSoftLimit()</a>	Configure the software limit.	
<a href="#">configMachinePPU()</a>	Configure the PPU (Pulse Per Unit).	
<a href="#">configMachineAxisDirection()</a>	Configure the motor direction.	
<a href="#">configMachinePositionFeedbackSource()</a>	Configure the position feedback source for the specified machine axis.	
<a href="#">configMachinePositionFeedbackScale()</a>	Configure the position feedback scale of the specified machine axis.	
<a href="#">configMachinePositionFeedbackOffset()</a>	Configure the position feedback offset of the specified machine axis.	
<a href="#">configMachineStepLossCompensationMode()</a>	Configure the Step Loss Compensation mode.	
<a href="#">configMachineStepLossCompensationMaxError()</a>	Configure the maximum position error for Step Loss Compensation.	
<a href="#">configMachineSelfStartingSpeed()</a>	Configure the Self-Starting Speed for the specified machine axis.	
<a href="#">configMachineG54WorkOffset()</a>	Configure the offset for the G54 work coordinate system.	
<a href="#">configMachineG55WorkOffset()</a>	Configure the offset for the G55 work coordinate system.	

<a href="#">configMachineG56WorkOffset()</a>	Configure the offset for the G56 work coordinate system.	
<a href="#">configMachineG57WorkOffset()</a>	Configure the offset for the G57 work coordinate system.	
<a href="#">configMachineG58WorkOffset()</a>	Configure the offset for the G58 work coordinate system.	
<a href="#">configMachineG59WorkOffset()</a>	Configure the offset for the G59 work coordinate system.	
<a href="#">configEncoderMode()</a>	Configure the encoder mode.	
<a href="#">configEncoderDigitalFilter()</a>	Configure the encoder digital filter.	
<a href="#">configEncoderRange()</a>	Configure the maximum encoder counter.	
<a href="#">configEncoderInputPolarity()</a>	Configure the polarity of the encoder input pins.	
<a href="#">configEncoderIndexReset()</a>	Enable or disable the encoder index signal reset counter function.	

- **Note 1:** The function can only be called in a callback function under some conditions.

## 2.3.6.1 Initialization Functions

Initialization-related functions for the EthercatDevice\_DmpStepper\_Generic class.

Functions:

- [attach\(\)](#)
- [detach\(\)](#)
- [cia402GetServo\(\)](#)

## attach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::attach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11MP3S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
    master.begin();
    device.attach(0, master);
    master.start();
}

void loop() {
    // ...
}
```

## detach()

### Description

This function behaves the same as [EthercatDevice\\_Generic::detach\(\)](#).

Please refer to that section for the detailed description, syntax, parameters, and return values.

### Example Code

For EthercatDevice\_QECR11MP3S.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  delay(3000);

  device.detach();
  master.end();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## cia402GetServo()

### Description

Obtain the EthercatDevice\_CiA402 object pointer of the specified motor on the EtherCAT SubDevice. If the `servo` parameter is `NULL`, the first call to this function will initialize an internal CiA402 object and return its pointer. Subsequent calls will only return the CiA402 object pointer. If the `servo` parameter is not `NULL` and the internal CiA402 object has been initialized, the internal object will be deinitialized and the input `servo` parameter will be used to initialize a new CiA402 object.

### Syntax

```
EthercatDevice_CiA402 *cia402GetServo(int motor, EthercatDevice_CiA402 *servo = NULL);
```

### Parameters

- `[in] motor`  
The specified motor number on the EtherCAT SubDevice:  
1: Motor 1.  
2: Motor 2.  
3: Motor 3.
- `[in] servo`  
The pointer to an object of the EthercatDevice\_CiA402 class declared by the user.  
The default is `NULL`.

### Return Value

Return an object pointer of the EthercatDevice\_CiA402 class.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function only works in CiA 402 on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

\*WARNING: The function can only be called within a callback function under the following conditions: it has been called before, and the 'servo' parameter is `NULL`.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;
EthercatDevice_CiA402 motor1;
EthercatDevice_CiA402 motor2;

EthercatDevice_CiA402 *pMotor[3];

void setup() {
```

```
master.begin();
device.attach(0, master);
pMotor[0] = device.cia402GetServo(1, &motor1);
pMotor[1] = device.cia402GetServo(2, &motor2);
pMotor[2] = device.cia402GetServo(3);
}

void loop() {
  // ...
}
```

## 2.3.6.2 Control Functions

Control functions for the EthercatDevice\_DmpStepper\_Generic class.

Functions:

- [update\(\)](#)
- [attachInterrupt\(\)](#)

## update()

### Description

Update state machines and internal variables for each function on the EtherCAT SubDevice.

### Syntax

```
int update();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.update();
  // ...
}
```

## attachInterrupt()

### Description

Register the event callback function for the EtherCAT SubDevice.

### Syntax

```
int attachInterrupt(void (*callback)(int));
```

### Parameters

- `[in] callback`

The event callback function to be registered has an integer-type parameter that indicates the event type. The supported event types are as follows:

Definition	Code	Description
ECAT_EMERGENCY_STOPPED	1	Emergency stop occurred.
ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED	2	The X-axis limit switch has been touched.
ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED	3	The Y-axis limit switch has been touched.
ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED	4	The Z-axis limit switch has been touched.

The remaining event types are determined using the following macros. Since the status of the following events are stored in the process data and this data is not latched, it is recommended to call [update\(\)](#) in the cyclic callback to handle these events and prevent event loss.

Definition	Description
IS_ECAT_ENCODER_1_INDEX_RESET(event)	The Index Reset event of Encoder 1 has been triggered.
IS_ECAT_ENCODER_2_INDEX_RESET(event)	The Index Reset event of Encoder 2 has been triggered.
IS_ECAT_ENCODER_3_INDEX_RESET(event)	The Index Reset event of Encoder 3 has been triggered.
IS_ECAT_ENCODER_X_INDEX_RESET(event)	The Index Reset event of Encoder X has been triggered.
IS_ECAT_ENCODER_Y_INDEX_RESET(event)	The Index Reset event of Encoder Y has been triggered.
IS_ECAT_ENCODER_Z_INDEX_RESET(event)	The Index Reset event of Encoder Z has been triggered.
IS_ECAT_ENCODER_1_OVERFLOW(event)	The Overflow event of Encoder 1 has been triggered.
IS_ECAT_ENCODER_2_OVERFLOW(event)	The Overflow event of Encoder 2 has been triggered.

IS_ECAT_ENCODER_3_OVERFLOW(event)	The Overflow event of Encoder 3 has been triggered.
IS_ECAT_ENCODER_X_OVERFLOW(event)	The Overflow event of Encoder X has been triggered.
IS_ECAT_ENCODER_Y_OVERFLOW(event)	The Overflow event of Encoder Y has been triggered.
IS_ECAT_ENCODER_Z_OVERFLOW(event)	The Overflow event of Encoder Z has been triggered.
IS_ECAT_ENCODER_1_UNDERFLOW(event)	The Underflow event of Encoder 1 has been triggered.
IS_ECAT_ENCODER_2_UNDERFLOW(event)	The Underflow event of Encoder 2 has been triggered.
IS_ECAT_ENCODER_3_UNDERFLOW(event)	The Underflow event of Encoder 3 has been triggered.
IS_ECAT_ENCODER_X_UNDERFLOW(event)	The Underflow event of Encoder X has been triggered.
IS_ECAT_ENCODER_Y_UNDERFLOW(event)	The Underflow event of Encoder Y has been triggered.
IS_ECAT_ENCODER_Z_UNDERFLOW(event)	The Underflow event of Encoder Z has been triggered.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is non-blocking and can be called in callback functions.

Since this callback function is invoked within the [update\(\)](#) function, if the [update\(\)](#) function is called in an interrupt callback function, specific usage restrictions must be adhered to. Please refer to [Callback Functions](#) for details.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

int emergency_stopped;
int x_limil_touched;
int y_limil_touched;
int z_limil_touched;
int encoder_index_reset[3];
int encoder_overflow[3];
int encoder_underflow[3];
```

```

int encoder_xyz_index_reset[3];
int encoder_xyz_overflow[3];
int encoder_xyz_underflow[3];

void Callback(int event) {
    switch (event) {
        case ECAT_EMERGENCY_STOPPED:
            emergency_stopped = 1;
            return;
        case ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED:
            x_limil_touched = 1;
            return;
        case ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED:
            y_limil_touched = 1;
            return;
        case ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED:
            z_limil_touched = 1;
            return;
        default:
            break;
    }

    if (IS_ECAT_ENCODER_1_INDEX_RESET(event))
        encoder_index_reset[0] = 1;
    else if (IS_ECAT_ENCODER_2_INDEX_RESET(event))
        encoder_index_reset[1] = 1;
    else if (IS_ECAT_ENCODER_3_INDEX_RESET(event))
        encoder_index_reset[2] = 1;
    else if (IS_ECAT_ENCODER_1_OVERFLOW(event))
        encoder_overflow[0] = 1;
    else if (IS_ECAT_ENCODER_2_OVERFLOW(event))
        encoder_overflow[1] = 1;
    else if (IS_ECAT_ENCODER_3_OVERFLOW(event))
        encoder_overflow[2] = 1;
    else if (IS_ECAT_ENCODER_1_UNDERFLOW(event))
        encoder_underflow[0] = 1;
    else if (IS_ECAT_ENCODER_2_UNDERFLOW(event))
        encoder_underflow[1] = 1;
    else if (IS_ECAT_ENCODER_3_UNDERFLOW(event))
        encoder_underflow[2] = 1;

    if (IS_ECAT_ENCODER_X_INDEX_RESET(event))

```

```

    encoder_xyz_index_reset[0] = 1;
else if (IS_ECAT_ENCODER_Y_INDEX_RESET(event))
    encoder_xyz_index_reset[1] = 1;
else if (IS_ECAT_ENCODER_Z_INDEX_RESET(event))
    encoder_xyz_index_reset[2] = 1;
else if (IS_ECAT_ENCODER_X_OVERFLOW(event))
    encoder_xyz_overflow[0] = 1;
else if (IS_ECAT_ENCODER_Y_OVERFLOW(event))
    encoder_xyz_overflow[1] = 1;
else if (IS_ECAT_ENCODER_Z_OVERFLOW(event))
    encoder_xyz_overflow[2] = 1;
else if (IS_ECAT_ENCODER_X_UNDERFLOW(event))
    encoder_xyz_underflow[0] = 1;
else if (IS_ECAT_ENCODER_Y_UNDERFLOW(event))
    encoder_xyz_underflow[1] = 1;
else if (IS_ECAT_ENCODER_Z_UNDERFLOW(event))
    encoder_xyz_underflow[2] = 1;
}

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);
    device.attachInterrupt(Callback);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}

void loop() {
    if (emergency_stopped) {
        emergency_stopped = 0;
        Serial.println("ECAT_EMERGENCY_STOPPED");
    }
    if (x_limil_touched) {
        x_limil_touched = 0;
        Serial.println("ECAT_MACHINE_X_AXIS_LIMIT_TOUCHED");
    }
}

```

```
if (y_limil_touched) {
  y_limil_touched = 0;
  Serial.println("ECAT_MACHINE_Y_AXIS_LIMIT_TOUCHED");
}
if (z_limil_touched) {
  z_limil_touched = 0;
  Serial.println("ECAT_MACHINE_Z_AXIS_LIMIT_TOUCHED");
}
if (encoder_index_reset[0]) {
  encoder_index_reset[0] = 0;
  Serial.println("IS_ECAT_ENCODER_1_INDEX_RESET");
}
if (encoder_index_reset[1]) {
  encoder_index_reset[1] = 0;
  Serial.println("IS_ECAT_ENCODER_2_INDEX_RESET");
}
if (encoder_index_reset[2]) {
  encoder_index_reset[2] = 0;
  Serial.println("IS_ECAT_ENCODER_3_INDEX_RESET");
}
if (encoder_overflow[0]) {
  encoder_overflow[0] = 0;
  Serial.println("IS_ECAT_ENCODER_1_OVERFLOW");
}
if (encoder_overflow[1]) {
  encoder_overflow[1] = 0;
  Serial.println("IS_ECAT_ENCODER_2_OVERFLOW");
}
if (encoder_overflow[2]) {
  encoder_overflow[2] = 0;
  Serial.println("IS_ECAT_ENCODER_3_OVERFLOW");
}
if (encoder_underflow[0]) {
  encoder_underflow[0] = 0;
  Serial.println("IS_ECAT_ENCODER_1_UNDERFLOW");
}
if (encoder_underflow[1]) {
  encoder_underflow[1] = 0;
  Serial.println("IS_ECAT_ENCODER_2_UNDERFLOW");
}
if (encoder_underflow[2]) {
  encoder_underflow[2] = 0;
}
```

```
    Serial.println("IS_ECAT_ENCODER_3_UNDERFLOW");
}
if (encoder_xyz_index_reset[0]) {
    encoder_xyz_index_reset[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_INDEX_RESET");
}
if (encoder_xyz_index_reset[1]) {
    encoder_xyz_index_reset[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_INDEX_RESET");
}
if (encoder_xyz_index_reset[2]) {
    encoder_xyz_index_reset[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_INDEX_RESET");
}
if (encoder_xyz_overflow[0]) {
    encoder_xyz_overflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_OVERFLOW");
}
if (encoder_xyz_overflow[1]) {
    encoder_xyz_overflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_OVERFLOW");
}
if (encoder_xyz_overflow[2]) {
    encoder_xyz_overflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_OVERFLOW");
}
if (encoder_xyz_underflow[0]) {
    encoder_xyz_underflow[0] = 0;
    Serial.println("IS_ECAT_ENCODER_X_UNDERFLOW");
}
if (encoder_xyz_underflow[1]) {
    encoder_xyz_underflow[1] = 0;
    Serial.println("IS_ECAT_ENCODER_Y_UNDERFLOW");
}
if (encoder_xyz_underflow[2]) {
    encoder_xyz_underflow[2] = 0;
    Serial.println("IS_ECAT_ENCODER_Z_UNDERFLOW");
}
// ...
}
```

## 2.3.6.3 Machine Functions

Machine functions for the EthercatDevice\_DmpStepper\_Generic class.

Functions:

- [machineEnableSoftLimit\(\)](#)
- [machineDisableSoftLimit\(\)](#)
- [machineIsEmergencyStopped\(\)](#)
- [machineSetEmergencyStop\(\)](#)
- [machineClearEmergencyStop\(\)](#)
- [machineIsLimitTouched\(\)](#)
- [machineIsHomingAttained\(\)](#)
- [machineIsServoOn\(\)](#)
- [machineIsMoving\(\)](#)
- [machineIsPositionErrorExceeded\(\)](#)
- [machineServoOn\(\)](#)
- [machineServoOff\(\)](#)
- [machineSetHomingSpeed\(\)](#)
- [machineStartHoming\(\)](#)
- [machineGcode\(\)](#)
- [machineActualPosition\(\)](#)

## machineEnableSoftLimit()

### Description

Enable the software limit function for the machine on the EtherCAT SubDevice. For more information about software limits, please refer to [configMachineSoftLimit\(\)](#).

### Syntax

```
int machineEnableSoftLimit();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#). This function does not work in the following case:

- The machine is servo-off.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    Serial.begin(115200);

    master.begin();
    device.attach(0, master);

    device.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, -50, 50);
    device.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, -50, 50);
    device.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, -50, 50);

    master.attachCyclicCallback(CyclicCallback);
    master.start();
}
```

```
device.machineServoOn();
while (device.machineIsServoOn() == 0) {}

device.machineEnableSoftLimit();
device.machineGcode("G1 X-100 Y-100 Z-100");
device.machineGcode("G1 X100 Y100 Z100");
delay(10);

while (device.machineIsMoving()) {
  Serial.print("Actual Position => ");
  Serial.print("X:");
  Serial.print(device.machineActualPosition(ECAT_MACHINE_X_AXIS), 2);
  Serial.print(", Y:");
  Serial.print(device.machineActualPosition(ECAT_MACHINE_Y_AXIS), 2);
  Serial.print(", Z:");
  Serial.println(device.machineActualPosition(ECAT_MACHINE_Z_AXIS), 2);
  delay(10);
  // ...
}

device.machineServoOff();
while (device.machineIsServoOn() == 1) {}

master.stop();
master.end();
}

void loop() {
}
```

## machineDisableSoftLimit()

### Description

Disable the software limit function for the machine on the EtherCAT SubDevice. For more information about software limits, please refer to [configMachineSoftLimit\(\)](#).

### Syntax

```
int machineDisableSoftLimit();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#). This function does not work in the following case:

- The machine is servo-off.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  device.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, -50, 50);
  device.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, -50, 50);
  device.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, -50, 50);

  master.attachCyclicCallback(CyclicCallback);
  master.start();
```

```
device.machineServoOn();
while (device.machineIsServoOn() == 0) {}

device.machineDisableSoftLimit();
device.machineGcode("G1 X-100 Y-100 Z-100");
device.machineGcode("G1 X100 Y100 Z100");
delay(10);

while (device.machineIsMoving()) {
  Serial.print("Actual Position => ");
  Serial.print("X:");
  Serial.print(device.machineActualPosition(ECAT_MACHINE_X_AXIS), 2);
  Serial.print(", Y:");
  Serial.print(device.machineActualPosition(ECAT_MACHINE_Y_AXIS), 2);
  Serial.print(", Z:");
  Serial.println(device.machineActualPosition(ECAT_MACHINE_Z_AXIS), 2);
  delay(10);
  // ...
}

device.machineServoOff();
while (device.machineIsServoOn() == 1) {}

master.stop();
master.end();
}

void loop() {
}
```

## machineIsEmergencyStopped()

### Description

Check if the machine on the EtherCAT SubDevice is emergency stopped. There are two primary conditions that can trigger the emergency stop:

- **Hardware emergency stop**

This occurs when the hardware emergency stop switch is physically activated. This switch is typically a physical button located on the machine. It is designed to halt the machine's operation in case of an immediate safety hazard.

- **User-initiated emergency stop**

This occurs when the user calls the [machineSetEmergencyStop\(\)](#) function. This function is typically used to manually activate the emergency stop state, often through a software interface or control panel.

### Syntax

```
int machineIsEmergencyStopped();
```

### Parameters

None.

### Return Value

Return whether the machine is emergency stopped. 1 means the Machine is in the Emergency Stop, and 0 means none.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}
```

```
void loop() {  
  Serial.print("Emergency Stopped: ");  
  Serial.println(device.machineIsEmergencyStopped());  
  delay(1000);  
  //...  
}
```

## machineSetEmergencyStop()

### Description

Initiate the emergency stop for the machine on the EtherCAT SubDevice. This function is typically used to manually activate the emergency stop state, often through a software interface or control panel.

### Syntax

```
int machineSetEmergencyStop();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
  master.start();

  device.machineSetEmergencyStop();
  delay(10);
  Serial.print("Emergency Stopped: ");
  Serial.println(device.machineIsEmergencyStopped());
}

void loop() {
  // ...
}
```

## machineClearEmergencyStop()

### Description

Clear the emergency stop for the machine on the EtherCAT SubDevice. To clear the emergency stop state and resume normal machine operation, ensure the physical hardware emergency stop switch is deactivated or reset to its normal position before calling this function.

Since this function is a non-blocking function and the [update\(\)](#) function needs to be called continuously to execute the state machine, it may take some time to complete, so the related status may take some time to respond, such as [machineIsEmergencyStopped\(\)](#).

### Syntax

```
int machineClearEmergencyStop();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.machineClearEmergencyStop();
}

void loop() {
  device.update();
  // ...
}
```

## machineIsLimitTouched()

### Description

Check if the limit switch of the specified machine axis on the EtherCAT SubDevice is touched.

### Syntax

```
int machineIsLimitTouched(int machine_axis);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

### Return Value

Return whether the limit switch of the specified machine axis is touched. 1 means the Machine Limit Switch is touched, and 0 means none.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
```

```
    master.start();
}

void loop() {
    Serial.print("Limit Switch => ");
    Serial.print("X: ");
    Serial.print(device.machineIsLimitTouched(ECAT_MACHINE_X_AXIS));
    Serial.print(", Y: ");
    Serial.print(device.machineIsLimitTouched(ECAT_MACHINE_Y_AXIS));
    Serial.print(", Z: ");
    Serial.println(device.machineIsLimitTouched(ECAT_MACHINE_Z_AXIS));
    delay(1000);
    // ...
}
```

## machineIsHomingAttained()

### Description

Check if the machine on the EtherCAT SubDevice is homing attained.

### Syntax

```
int machineIsHomingAttained();
```

### Parameters

None.

### Return Value

Return whether the machine is homing attained. 1 means the Machine is homing attained, and 0 means none.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.machineServoOn();
    while (device.machineIsServoOn() == 0);

    device.machineStartHoming();
    delay(10);
    while (!device.machineIsHomingAttained());
    // ..
```

```
device.machineServoOff();  
while (device.machineIsServoOn() == 1);  
}  
  
void loop() {  
  // ...  
}
```

## machineIsServoOn()

### Description

Check if the machine on the EtherCAT SubDevice is servo-on.

### Syntax

```
int machineIsServoOn();
```

### Parameters

None.

### Return Value

Return whether the machine is servo-on. 1 means the Machine is Servo-on, and 0 means none. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.machineServoOn();
    while (device.machineIsServoOn() == 0);
    delay(1000);
    // ..
    device.machineServoOff();
    while (device.machineIsServoOn() == 1);
}
```

```
void loop() {  
  // ...  
}
```

## machineIsMoving()

### Description

Check if the machine on the EtherCAT SubDevice is moving.

### Syntax

```
int machineIsMoving();
```

### Parameters

None.

### Return Value

Return whether the machine is moving. 1 means the Machine is moving, and 0 means none.

If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.machineServoOn();
    while (device.machineIsServoOn() == 0);

    device.machineGcode("G1 X0 Y0 Z0");
    device.machineGcode("G1 X100 Y50 Z25");
    delay(10);
    while (device.machineIsMoving());
    // ...
```

```
device.machineServoOff();  
while (device.machineIsServoOn() == 1);  
}  
  
void loop() {  
  // ...  
}
```

## machineIsPositionErrorExceeded()

### Description

Check if the position error of the specified machine axis has exceeded the maximum position error for Step Loss Compensation.

### Syntax

```
int machineIsPositionErrorExceeded(int machine_axis);
```

### Parameters

- [in] `machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

### Return Value

Return whether the position error has exceeded the maximum position error. If the return value is less than 0, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is non-blocking and can be called in callback functions.

This function only works in *G-code Controller* mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void Callback(int event) {
    if (event == ECAT_EMERGENCY_STOPPED) {
        if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_X_AXIS)) {
            // ...
        }
        if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_Y_AXIS)) {
            // ...
        }
        if (device.machineIsPositionErrorExceeded(ECAT_MACHINE_Z_AXIS)) {
```

```
    // ...
  }
}

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  device.attachInterrupt(Callback);
  master.attachCyclicCallback(CyclicCallback);
  master.start();
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## machineServoOn()

### Description

Turn on all motors of the machine on the EtherCAT SubDevice.

Since this function is a non-blocking function and the [update\(\)](#) function needs to be called continuously to execute the state machine, it may take some time to complete, so the related status may take some time to respond, such as [machineIsServoOn\(\)](#).

### Syntax

```
int machineServoOn();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.machineServoOn();
    while (device.machineIsServoOn() == 0);
    delay(1000);
    // ..
    device.machineServoOff();
}
```

```
while (device.machineIsServoOn() == 1);  
}  
  
void loop() {  
  // ...  
}
```

## machineServoOff()

### Description

Turn off all motors of the machine on the EtherCAT SubDevice.

Since this function is a non-blocking function and the [update\(\)](#) function needs to be called continuously to execute the state machine, it may take some time to complete, so the related status may take some time to respond, such as [machineIsServoOn\(\)](#).

### Syntax

```
int machineServoOff();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

This function does not work in the following cases:

- The machine is homing.
- The machine is moving.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();

    device.machineServoOn();
    while (device.machineIsServoOn() == 0);
```

```
delay(1000);  
// ..  
device.machineServoOff();  
while (device.machineIsServoOn() == 1);  
}  
  
void loop() {  
  // ...  
}
```

## machineSetHomingSpeed()

### Description

Set the homing speed of the specified machine axis on the EtherCAT SubDevice.

### Syntax

```
int machineSetHomingSpeed(int machine_axis, double mm_per_min);
```

### Parameters

- [in] `machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] `mm_per_min`

The desired homing speed in millimeters per minute.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    device.machineSetHomingSpeed(ECAT_MACHINE_X_AXIS, 3000);
```

```
device.machineSetHomingSpeed(ECAT_MACHINE_Y_AXIS, 3000);
device.machineSetHomingSpeed(ECAT_MACHINE_Z_AXIS, 3000);
master.attachCyclicCallback(CyclicCallback);
master.start();

device.machineServoOn();
while (device.machineIsServoOn() == 0);

device.machineStartHoming();
delay(10);
while (!device.machineIsHomingAttained());
// ..

device.machineServoOff();
while (device.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

## machineStartHoming()

### Description

Initiate the homing procedure for the machine on the EtherCAT SubDevice.

Since this function is a non-blocking function and the [update\(\)](#) function needs to be called continuously to execute the state machine, it may take some time to complete, so the related status may take some time to respond, such as [machineIsHomingAttained\(\)](#).

### Syntax

```
int machineStartHoming();
```

### Parameters

None.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in G-code Controller mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

This function does not work in the following cases:

- The machine is emergency stopped.
- The machine is servo-off.
- The machine is homing.
- The machine is moving.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}

void setup() {
    master.begin();
    device.attach(0, master);
    master.attachCyclicCallback(CyclicCallback);
    master.start();
}
```

```
device.machineServoOn();
while (device.machineIsServoOn() == 0);

device.machineStartHoming();
delay(10);
while (!device.machineIsHomingAttained());
// ..

device.machineServoOff();
while (device.machineIsServoOn() == 1);

}

void loop() {
  // ...
}
```

## machineGcode()

### Description

Send a G-code command to the EtherCAT SubDevice.

Since this function is a non-blocking function and the [update\(\)](#) function needs to be called continuously to execute the state machine, it may take some time to complete, so the related status may take some time to respond, such as [machineIsMoving\(\)](#).

### Syntax

```
int machineGcode(const char *fmt, ...);
```

### Parameters

- `[in] fmt`  
The string of the G-code command to be transmitted to the EtherCAT SubDevice. This is a pointer to a null-terminated string containing the format specification for the output. The format string follows the same format as the `printf` function in C, allowing for insertion of variables and formatting options.
- `[in] ...`  
This is a variable number of arguments that will be inserted into the formatted string according to the format specifiers in the `fmt` string.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in *G-code Controller* mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

This function does not work in the following cases:

- The machine is emergency stopped.
- The machine is servo-off.
- The machine is homing.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
    device.update();
}
```

```
void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  device.machineServoOn();
  while (device.machineIsServoOn() == 0);

  while (1) {
    device.machineGcode("G1 X100 Y50 Z25");
    delay(1000);
    device.machineGcode("G1 X0 Y0 Z0");
    delay(1000);
    // ...
  }

  device.machineServoOff();
  while (device.machineIsServoOn() == 1);
}

void loop() {
  // ...
}
```

## machineActualPosition()

### Description

Get the current position of the specified machine axis on the EtherCAT SubDevice.

### Syntax

```
double machineActualPosition(int machine_axis);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

### Return Value

Return the current position of the specified machine axis in millimeters.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

This function only works in *G-code Controller* mode on the EtherCAT SubDevice. For more details, please refer to [configDeviceMode\(\)](#).

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void CyclicCallback() {
  device.update();
}

void setup() {
  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(CyclicCallback);
  master.start();

  device.machineServoOn();
}
```

```
while (device.machineIsServoOn() == 0);
}

void loop() {
  if (device.machineIsMoving() == 0) {
    device.machineGcode("G1 X0 Y0 Z0");
    device.machineGcode("G1 X100 Y50 Z25");
    delay(10);
  }
  printf("Actual Position => ");
  printf("X:%.2f, ", device.machineActualPosition(ECAT_MACHINE_X_AXIS));
  printf("Y:%.2f, ", device.machineActualPosition(ECAT_MACHINE_Y_AXIS));
  printf("Z:%.2f\n", device.machineActualPosition(ECAT_MACHINE_Z_AXIS));
  delay(10);
  // ...
}
```

## 2.3.6.4 Encoder Functions

Encoder functions for the EthercatDevice\_DmpStepper\_Generic class.

Functions:

- [encoderWrite\(\)](#)
- [encoderDirectionRead\(\)](#)
- [encoderRead\(\)](#)

## encoderWrite()

### Description

Write the counter of the specified encoder on the EtherCAT SubDevice.

### Syntax

```
int encoderWrite(int encoder, int32_t value);
```

### Parameters

- [in] encoder

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by

[configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- [in] value

The value to be written.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);
  device.encoderWrite(ECAT_ENCODER_1, 100);
  device.encoderWrite(ECAT_ENCODER_2, 100);
  device.encoderWrite(ECAT_ENCODER_3, 100);
  master.start();
}
```

```
void loop() {  
  // ...  
}
```

## encoderDirectionRead()

### Description

Read the current direction of the specified encoder on the EtherCAT SubDevice.

### Syntax

```
int encoderDirectionRead(int encoder);
```

### Parameters

- [in] encoder

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by

[configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

### Return Value

Return the current direction of the specified encoder. A return value of 0 indicates forward rotation, while a return value of 1 indicates reverse rotation. If the returned value is less than zero, it indicates an [error code](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Encoder 1 Direction: ");
```

```
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_1));
Serial.print("Encoder 2 Direction: ");
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_2));
Serial.print("Encoder 3 Direction: ");
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_3));
Serial.print("Encoder X Direction: ");
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_X));
Serial.print("Encoder Y Direction: ");
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_Y));
Serial.print("Encoder Z Direction: ");
Serial.println(device.encoderDirectionRead(ECAT_ENCODER_Z));
delay(1000);
// ...
}
```

## encoderRead()

### Description

Read the counter of the specified encoder on the EtherCAT SubDevice.

### Syntax

```
int32_t encoderRead(int encoder);
```

### Parameters

- [in] encoder

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by

[configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

### Return Value

Return the counter of the specified encoder with 32-bit signed integer.

### Comment

This function must be called after a successful execution of [EthercatMaster::start\(\)](#) and before [EthercatMaster::stop\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Encoder 1: ");
  Serial.println(device.encoderRead(ECAT_ENCODER_1));
  Serial.print("Encoder 2: ");
```

```
Serial.println(device.encoderRead(ECAT_ENCODER_2));  
Serial.print("Encoder 3: ");  
Serial.println(device.encoderRead(ECAT_ENCODER_3));  
Serial.print("Encoder X: ");  
Serial.println(device.encoderRead(ECAT_ENCODER_X));  
Serial.print("Encoder Y: ");  
Serial.println(device.encoderRead(ECAT_ENCODER_Y));  
Serial.print("Encoder Z: ");  
Serial.println(device.encoderRead(ECAT_ENCODER_Z));  
delay(1000);  
// ...  
}
```

## 2.3.6.5 Configuration Functions

Configuration functions for the EthercatDevice\_DmpStepper\_Generic class.

Functions:

- [getDeviceMode\(\)](#)
- [configDeviceMode\(\)](#)
- [configCiA402MotorResolution\(\)](#)
- [configCiA402MotorPositionFeedbackSource\(\)](#)
- [configCiA402MotorPositionFeedbackScale\(\)](#)
- [configCiA402MotorPositionFeedbackOffset\(\)](#)
- [configCiA402MotorSelfStartingSpeed\(\)](#)
- [configMachineAxisMapping\(\)](#)
- [configMachineDefaultFeedrate\(\)](#)
- [configMachineDefaultHomingSpeed\(\)](#)
- [configMachineHomingDirection\(\)](#)
- [configMachineHomingPriority\(\)](#)
- [configMachineMaxVelocity\(\)](#)
- [configMachineMaxAcceleration\(\)](#)
- [configMachineSoftLimit\(\)](#)
- [configMachinePPU\(\)](#)
- [configMachineAxisDirection\(\)](#)
- [configMachinePositionFeedbackSource\(\)](#)
- [configMachinePositionFeedbackScale\(\)](#)
- [configMachinePositionFeedbackOffset\(\)](#)
- [configMachineStepLossCompensationMode\(\)](#)
- [configMachineStepLossCompensationMaxError\(\)](#)
- [configMachineSelfStartingSpeed\(\)](#)
- [configMachineG54WorkOffset\(\)](#)
- [configMachineG55WorkOffset\(\)](#)
- [configMachineG56WorkOffset\(\)](#)
- [configMachineG57WorkOffset\(\)](#)
- [configMachineG58WorkOffset\(\)](#)
- [configMachineG59WorkOffset\(\)](#)
- [configEncoderMode\(\)](#)
- [configEncoderDigitalFilter\(\)](#)
- [configEncoderRange\(\)](#)
- [configEncoderInputPolarity\(\)](#)
- [configEncoderIndexReset\(\)](#)

## getDeviceMode()

### Description

Get the device mode for the EtherCAT SubDevice.

### Syntax

```
int getDeviceMode();
```

### Parameters

None.

### Return Value

Return the device mode. If the return value is less than 0, it indicates an [error code](#). For details about device modes, please refer to [configDeviceMode\(\)](#).

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  Serial.begin(115200);
  master.begin();
  device.attach(0, master);

  Serial.print("Device Mode: ");
  Serial.println(device.getDeviceMode());
}

void loop() {
  // ...
}
```

## configDeviceMode()

### Description

Configure the device mode for the EtherCAT SubDevice. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program. After calling this function, the device mode of the EtherCAT SubDevice will not change immediately. The EtherCAT SubDevice must be repowered for the change to take effect.

This EtherCAT SubDevice supports the following two device modes:

- **CiA 402 Servo**

This EtherCAT SubDevice is equipped with 3-axis CiA 402 stepper motors. Users can individually control each CiA 402 stepper motor.

- **G-code Controller**

This EtherCAT SubDevice functions as a G-code controller, responsible for parsing and executing G-code instructions.

### Syntax

```
int configDeviceMode(uint8_t mode);
```

### Parameters

- `[in] mode`

The device modes supported by this EtherCAT SubDevice are as follows:

Definition	Value	Description
ECAT_CIA402_SERVO_MODE	0	CiA 402 Servo mode.
ECAT_GCODE_CONTROLLER_MODE	1	G-code Controller mode.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configDeviceMode(ECAT_GCODE_CONTROLLER_MODE);
}
```

```
void loop() {  
  // ...  
}
```

## configCiA402MotorResolution()

### Description

Configure the motor resolution for the specified motor on the EtherCAT SubDevice in CiA 402 Servo mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configCiA402MotorResolution(int motor, uint32_t steps_per_rev);
```

### Parameters

- `[in] motor`

The specified motor number on the EtherCAT SubDevice:

Value	Description
1	The motor 1 on the EtherCAT SubDevice.
2	The motor 2 on the EtherCAT SubDevice.
3	The motor 3 on the EtherCAT SubDevice.

- `[in] steps_per_rev`

The motor resolution to be configured, specified in terms of steps per revolution.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorResolution(1, 3200);
  device.configCiA402MotorResolution(2, 3200);
  device.configCiA402MotorResolution(3, 3200);
  // ...
}

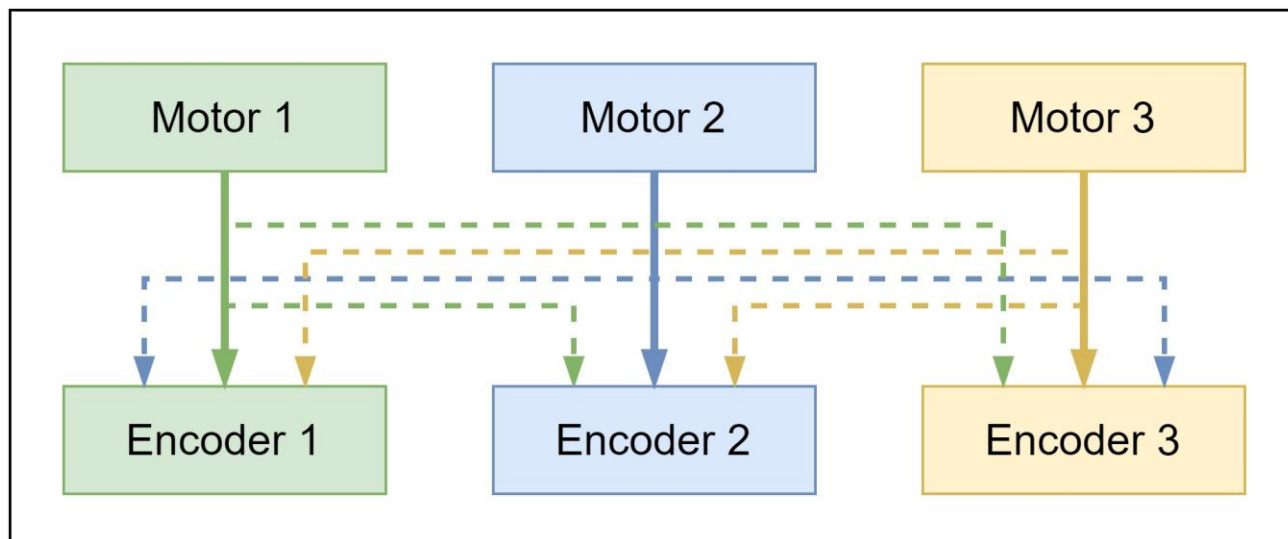
void loop() {
```

```
// ...  
}
```

## configCiA402MotorPositionFeedbackSource()

### Description

Configure the Position Feedback Source for the specified motor on the EtherCAT SubDevice.



### Syntax

```
int configCiA402MotorPositionFeedbackSource(int motor, int encoder);
```

### Parameters

- [in] motor

The specified motor number on the EtherCAT SubDevice:

Value	Description
1	The motor 1 on the EtherCAT SubDevice.
2	The motor 2 on the EtherCAT SubDevice.
3	The motor 3 on the EtherCAT SubDevice.

- [in] encoder

The position feedback source to be configured:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
Other Value	...	Disabled.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

**Example Code**

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackSource(1, ECAT_ENCODER_1);
  device.configCiA402MotorPositionFeedbackSource(2, ECAT_ENCODER_2);
  device.configCiA402MotorPositionFeedbackSource(3, ECAT_ENCODER_3);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configCiA402MotorPositionFeedbackScale()

### Description

Configure the Position Feedback Scale for the specified motor on the EtherCAT SubDevice.

Position Feedback = ( Encoder Raw Data × Scale ) + Offset

The *Scale* term represents a scaling factor that converts the digital encoder readings into real-world physical units. Essentially, it establishes a relationship between the number of counts from the encoder and the corresponding physical displacement or rotation.

This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configCiA402MotorPositionFeedbackScale(int motor, double scale);
```

### Parameters

- `[in] motor`

The specified motor number on the EtherCAT SubDevice:

Value	Description
1	The motor 1 on the EtherCAT SubDevice.
2	The motor 2 on the EtherCAT SubDevice.
3	The motor 3 on the EtherCAT SubDevice.

- `[in] scale`

The position feedback scale to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackScale(1, 2.0);
  device.configCiA402MotorPositionFeedbackScale(2, 2.0);
  device.configCiA402MotorPositionFeedbackScale(3, 2.0);
}
```

```
// ...  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

## configCiA402MotorPositionFeedbackOffset()

### Description

Configure the *Position Feedback Offset* for the specified motor on the EtherCAT SubDevice.

Position Feedback = ( Encoder Raw Data × Scale ) + Offset

The Offset term represents the initial displacement or starting point of a system relative to a reference point. It's essentially a correction factor that accounts for any discrepancies between the encoder's zero reading and the actual physical zero position of the system.

This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configCiA402MotorPositionFeedbackOffset(int motor, double offset);
```

### Parameters

- [in] motor

The specified motor number on the EtherCAT SubDevice:

Value	Description
1	The motor 1 on the EtherCAT SubDevice.
2	The motor 2 on the EtherCAT SubDevice.
3	The motor 3 on the EtherCAT SubDevice.

- [in] offset

The position feedback offset to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorPositionFeedbackOffset(1, 10.0);
  device.configCiA402MotorPositionFeedbackOffset(2, 10.0);
  device.configCiA402MotorPositionFeedbackOffset(3, 10.0);
}
```

```
// ...  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

## configCiA402MotorSelfStartingSpeed()

### Description

Configure the Self-Starting Speed for the specified motor on the EtherCAT SubDevice. The Self-Starting Speed indicates the highest stepping pulse frequency at which a stepper motor can instantaneously accelerate from a standstill to a stable running speed without using any acceleration ramp. If this speed is exceeded, the motor may lose steps or fail to start.

### Syntax

```
int configCiA402MotorSelfStartingSpeed(int motor, double rev_per_min);
```

### Parameters

- `[in] motor`

The specified motor number on the EtherCAT SubDevice:

Value	Description
1	The motor 1 on the EtherCAT SubDevice.
2	The motor 2 on the EtherCAT SubDevice.
3	The motor 3 on the EtherCAT SubDevice.

- `[in] rev_per_min`

The Self-Starting Speed to be configured is in RPM (Revolutions Per Minute)

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configCiA402MotorSelfStartingSpeed(1, 60.0);
  device.configCiA402MotorSelfStartingSpeed(2, 60.0);
  device.configCiA402MotorSelfStartingSpeed(3, 60.0);
  // ...
}

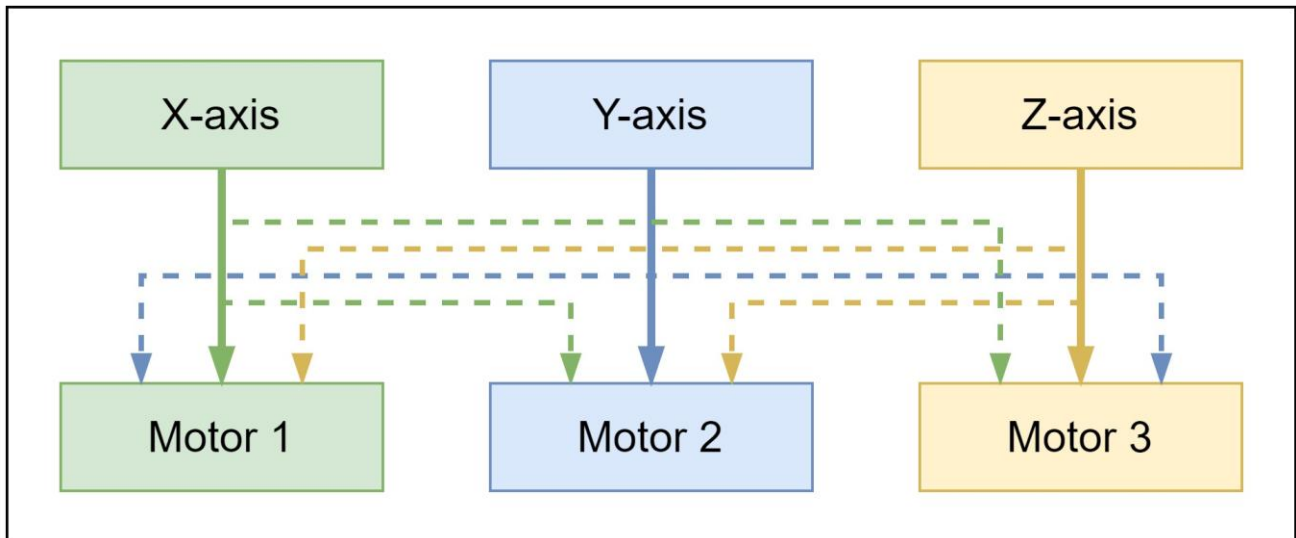
void loop() {
```

```
// put your main code here, to run repeatedly:  
  
}
```

## configMachineAxisMapping()

### Description

Configure the mapping between mechanical axes and motors for the EtherCAT SubDevice in G-code Controller mode.



This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program. After calling this function, the mapping will not change immediately. The EtherCAT SubDevice must be repowered for the change to take effect.

### Syntax

```
int configMachineAxisMapping(uint8_t mapping);
```

### Parameters

- `[in] mapping`  
The mapping of the mechanical axis to be configured. The EtherCAT SubDevice offers the following mapping options:

Code	X-axis	Y-axis	Z-axis
0	Motor 1	Motor 2	Motor 3
1	Motor 1	Motor 3	Motor 2
2	Motor 2	Motor 1	Motor 3
3	Motor 2	Motor 3	Motor 1
4	Motor 3	Motor 1	Motor 2
5	Motor 3	Motor 2	Motor 1

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

**Comment**

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

**Example Code**

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineAxisMapping(0);
}

void loop() {
  // ...
}
```

## configMachineDefaultFeedrate()

### Description

Configure the default feed rate for the EtherCAT SubDevice in G-code Controller mode. If the user has not yet specified a feed rate in a G-code movement command sent through [machineGcode\(\)](#), this default feed rate will be used. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineDefaultFeedrate(double mm_per_min);
```

### Parameters

- `[in] mm_per_min`

The default feed rate to be configured is in millimeters per minute.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineDefaultFeedrate(3000);
  // ...
}

void loop() {
  // ...
}
```

## configMachineDefaultHomingSpeed()

### Description

Configure the default homing speed of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This default homing speed will be used for homing operations if the user has not yet specified a homing speed for each axis using the [machineSetHomingSpeed\(\)](#) function. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineDefaultHomingSpeed(int machine_axis, double mm_per_min);
```

### Parameters

- [in] `machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] `mm_per_min`

The default homing speed to be configured is in millimeters per minute.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineDefaultHomingSpeed(ECAT_MACHINE_X_AXIS, 1000);
  device.configMachineDefaultHomingSpeed(ECAT_MACHINE_Y_AXIS, 1000);
  device.configMachineDefaultHomingSpeed(ECAT_MACHINE_Z_AXIS, 1000);
}
```

```
// ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachineHomingDirection()

### Description

Configure the homing direction of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineHomingDirection(int machine_axis, bool positive);
```

### Parameters

- `[in] machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- `[in] positive`

A Boolean value used to configure the homing direction.

- `true`: Search for the home switch in the **positive** direction.
- `false`: Search for the home switch in the **negative** direction.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineHomingDirection(ECAT_MACHINE_X_AXIS, false);
  device.configMachineHomingDirection(ECAT_MACHINE_Y_AXIS, false);
}
```

```
device.configMachineHomingDirection(ECAT_MACHINE_Z_AXIS, false);  
  // ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachineHomingPriority()

### Description

Configure the homing priority of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineHomingPriority(int machine_axis, uint8_t priority);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] priority

The homing priority to be configured. The value range is 0 to 255, and the lower the value, the higher the priority. If the values are equal, the priority order is not guaranteed.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineHomingPriority(ECAT_MACHINE_X_AXIS, 1);
  device.configMachineHomingPriority(ECAT_MACHINE_Y_AXIS, 2);
  device.configMachineHomingPriority(ECAT_MACHINE_Z_AXIS, 3);
}
```

```
// ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachineMaxVelocity()

### Description

Configure the maximum velocity of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. The purpose of this parameter is primarily to limit maximum speed of the motor to prevent exceeding the motor's specifications. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineMaxVelocity(int machine_axis, double mm_per_sec);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] mm\_per\_sec

The maximum velocity to be configured is in millimeters per second.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineMaxVelocity(ECAT_MACHINE_X_AXIS, 300);
  device.configMachineMaxVelocity(ECAT_MACHINE_Y_AXIS, 300);
  device.configMachineMaxVelocity(ECAT_MACHINE_Z_AXIS, 300);
}
```

```
// ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachineMaxAcceleration()

### Description

Configure the maximum acceleration for the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineMaxAcceleration(double mm_per_sec2);
```

### Parameters

- [in] mm\_per\_sec2

The maximum acceleration to be configured is in millimeters per second squared.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineMaxAcceleration(100000);
  // ...
}

void loop() {
  // ...
}
```

## configMachineSoftLimit()

### Description

Configure the software limit of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. These parameters are written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure these parameters each time before running the program.

Upon receiving a movement command from the user calling [machineGcode\(\)](#), the EtherCAT SubDevice internally calculates the target position for each axis.

- If the target position  $P$  for an axis is greater than the maximum position value  $P_{\max}$  for that axis, then  $P$  will be adjusted to  $P_{\max}$ .
- If the target position  $P$  for an axis is less than the minimum position value  $P_{\min}$  for that axis, then  $P$  will be adjusted to  $P_{\min}$ .

### Syntax

```
int configMachineSoftLimit(int machine_axis, double min, double max);
```

### Parameters

- `[in] machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- `[in] min`

The minimum position value  $P_{\min}$  of the software limit to be configured, in millimeters.

- `[in] max`

The maximum position value  $P_{\max}$  of the software limit to be configured, in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;
```

```
void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineSoftLimit(ECAT_MACHINE_X_AXIS, 0, 100);
  device.configMachineSoftLimit(ECAT_MACHINE_Y_AXIS, 0, 100);
  device.configMachineSoftLimit(ECAT_MACHINE_Z_AXIS, 0, 100);
}

void loop() {
  // ...
}
```

## configMachinePPU()

### Description

Configure the PPU (Pulse Per Unit) of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter defines the relationship between the movement of the machine and the motors. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachinePPU(int machine_axis, double pulses_per_mm);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] pulses\_per\_mm

The PPU to be configured is in pulses per millimeter.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachinePPU(ECAT_MACHINE_X_AXIS, 100);
  device.configMachinePPU(ECAT_MACHINE_Y_AXIS, 100);
  device.configMachinePPU(ECAT_MACHINE_Z_AXIS, 100);
}
```

```
// ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachineAxisDirection()

### Description

Configure the motor direction of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineAxisDirection(int machine_axis, int invert);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] invert

The motor direction to be configured. A value of 0 indicates normal direction, while any other value indicates inverted direction.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

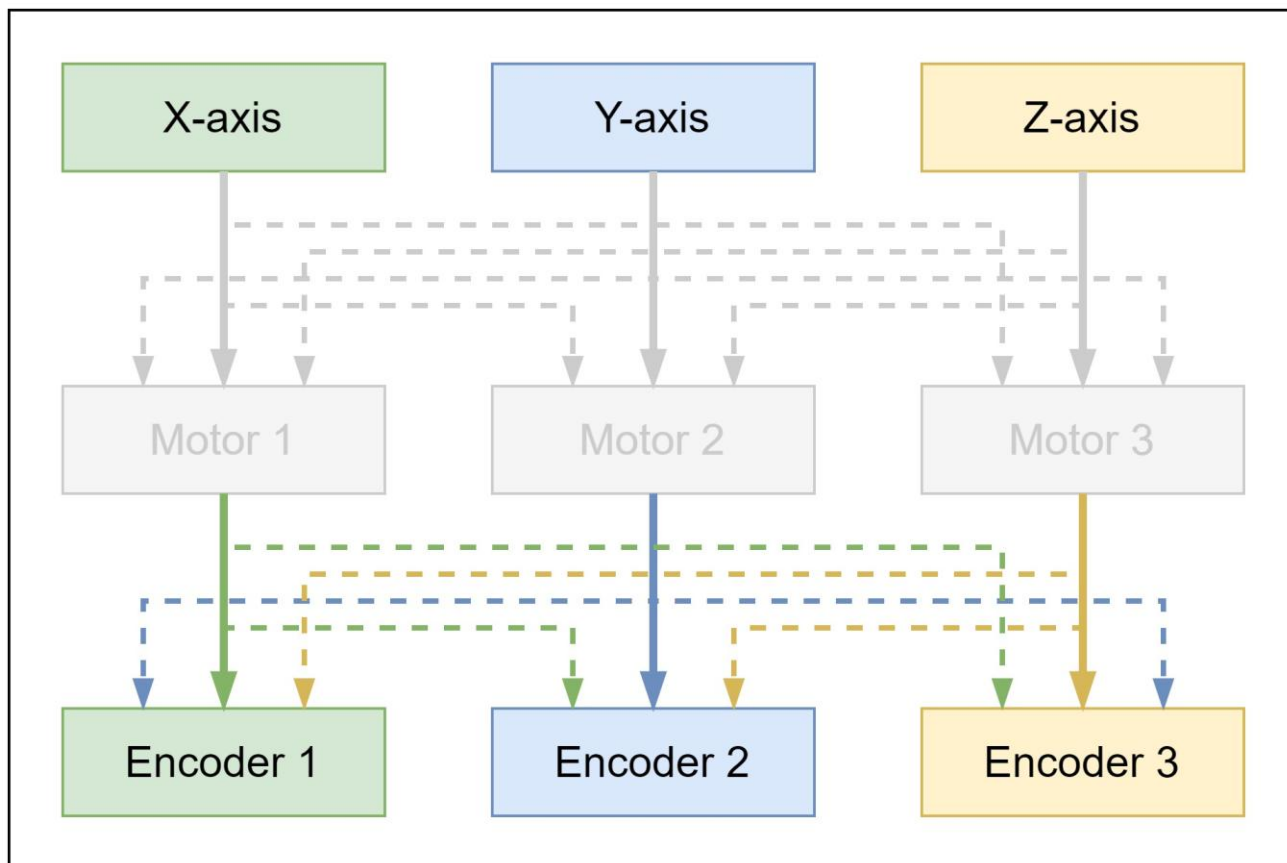
  device.configMachineAxisDirection(ECAT_MACHINE_X_AXIS, 0);
  device.configMachineAxisDirection(ECAT_MACHINE_Y_AXIS, 0);
  device.configMachineAxisDirection(ECAT_MACHINE_Z_AXIS, 0);
}
```

```
// ...  
}  
  
void loop() {  
  // ...  
}
```

## configMachinePositionFeedbackSource()

### Description

Configure the Position Feedback Source for the specified machine axis on the EtherCAT SubDevice.



### Syntax

```
int configMachinePositionFeedbackSource(int machine_axis, int encoder);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- `[in] encoder`

The position feedback source to be configured:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
Other Value	...	Disabled.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of `EthercatMaster::begin()`. This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachinePositionFeedbackSource(ECAT_MACHINE_X_AXIS,
  ECAT_ENCODER_1);
  device.configMachinePositionFeedbackSource(ECAT_MACHINE_Y_AXIS,
  ECAT_ENCODER_2);
  device.configMachinePositionFeedbackSource(ECAT_MACHINE_Z_AXIS,
  ECAT_ENCODER_3);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachinePositionFeedbackScale()

### Description

Configure the Position Feedback Scale of the specified machine axis on the EtherCAT SubDevice.

Position Feedback = ( Encoder Raw Data × Scale ) + Offset

The Scale term represents a scaling factor that converts the digital encoder readings into real-world physical units. Essentially, it establishes a relationship between the number of counts from the encoder and the corresponding physical displacement or rotation.

This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachinePositionFeedbackScale(int machine_axis, double scale);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] scale

The position feedback scale to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);
}
```

```
device.configMachinePositionFeedbackScale(ECAT_MACHINE_X_AXIS, 2.0);
device.configMachinePositionFeedbackScale(ECAT_MACHINE_Y_AXIS, 2.0);
device.configMachinePositionFeedbackScale(ECAT_MACHINE_Z_AXIS, 2.0);
// ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachinePositionFeedbackOffset()

### Description

Configure the Position Feedback Offset of the specified machine axis on the EtherCAT SubDevice.

Position Feedback = ( Encoder Raw Data × Scale ) + Offset

The Offset term represents the initial displacement or starting point of a system relative to a reference point. It's essentially a correction factor that accounts for any discrepancies between the encoder's zero reading and the actual physical zero position of the system.

This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachinePositionFeedbackOffset(int machine_axis, double offset);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] offset

The position feedback offset to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);
}
```

```
device.configMachinePositionFeedbackOffset(ECAT_MACHINE_X_AXIS, 10.0);
device.configMachinePositionFeedbackOffset(ECAT_MACHINE_Y_AXIS, 10.0);
device.configMachinePositionFeedbackOffset(ECAT_MACHINE_Z_AXIS, 10.0);
// ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineStepLossCompensationMode()

### Description

Configure the mode of Step Loss Compensation of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineStepLossCompensationMode(int machine_axis, uint8_t mode);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] mode

The mode of Step Loss Compensation to be configured.

- 0 for disable Step Loss Compensation.
- 1 for enable Step Loss Compensation.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineStepLossCompensationMode(ECAT_MACHINE_X_AXIS, 1);
  device.configMachineStepLossCompensationMode(ECAT_MACHINE_Y_AXIS, 1);
}
```

```
device.configMachineStepLossCompensationMode(ECAT_MACHINE_Z_AXIS, 1);  
// ...  
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

## configMachineStepLossCompensationMaxError()

### Description

Configure the maximum position error for Step Loss Compensation of the specified machine axis on the EtherCAT SubDevice in G-code Controller mode. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configMachineStepLossCompensationMaxError(int machine_axis, double max_error);
```

### Parameters

- [in] machine\_axis

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- [in] max\_error

The maximum position error for Step Loss Compensation to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_X_AXIS,
  20.0);
```

```
    device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_Y_AXIS,  
20.0);  
    device.configMachineStepLossCompensationMaxError(ECAT_MACHINE_Z_AXIS,  
20.0);  
    // ...  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
  
}
```

## configMachineSelfStartingSpeed()

### Description

Configure the Self-Starting Speed for the specified machine axis on the EtherCAT SubDevice. The Self-Starting Speed indicates the highest stepping pulse frequency at which a stepper motor can instantaneously accelerate from a standstill to a stable running speed without using any acceleration ramp. If this speed is exceeded, the motor may lose steps or fail to start.

### Syntax

```
int configMachineSelfStartingSpeed(int machine_axis, double rev_per_min);
```

### Parameters

- `[in] machine_axis`

The specified machine axis number on the EtherCAT SubDevice:

Definition	Value	Description
ECAT_MACHINE_X_AXIS	0	X-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Y_AXIS	1	Y-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .
ECAT_MACHINE_Z_AXIS	2	Z-axis, which mapping is determined by the <a href="#">configMachineAxisMapping()</a> .

- `[in] rev_per_min`

The Self-Starting Speed to be configured is in RPM (Revolutions Per Minute)

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineSelfStartingSpeed(ECAT_MACHINE_X_AXIS, 60.0);
  device.configMachineSelfStartingSpeed(ECAT_MACHINE_Y_AXIS, 60.0);
  device.configMachineSelfStartingSpeed(ECAT_MACHINE_Z_AXIS, 60.0);
  // ...
}
```

```
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

## configMachineG54WorkOffset()

### Description

Configure the offset for the G54 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG54WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG54WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineG55WorkOffset()

### Description

Configure the offset for the G55 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG55WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG55WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineG56WorkOffset()

### Description

Configure the offset for the G56 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG56WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG56WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineG57WorkOffset()

### Description

Configure the offset for the G57 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG57WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG57WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineG58WorkOffset()

### Description

Configure the offset for the G58 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG58WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG58WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configMachineG59WorkOffset()

### Description

Configure the offset for the G59 work coordinate system relative to the machine coordinate system in G-code Controller mode for the EtherCAT SubDevice.

### Syntax

```
int configMachineG59WorkOffset(double x_offset, double y_offset, double z_offset);
```

### Parameters

- `[in] x_offset`  
The X-axis work offset to be configured in millimeters.
- `[in] y_offset`  
The Y-axis work offset to be configured in millimeters.
- `[in] z_offset`  
The Z-axis work offset to be configured in millimeters.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configMachineG59WorkOffset(10.0, 10.0, 10.0);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configEncoderMode()

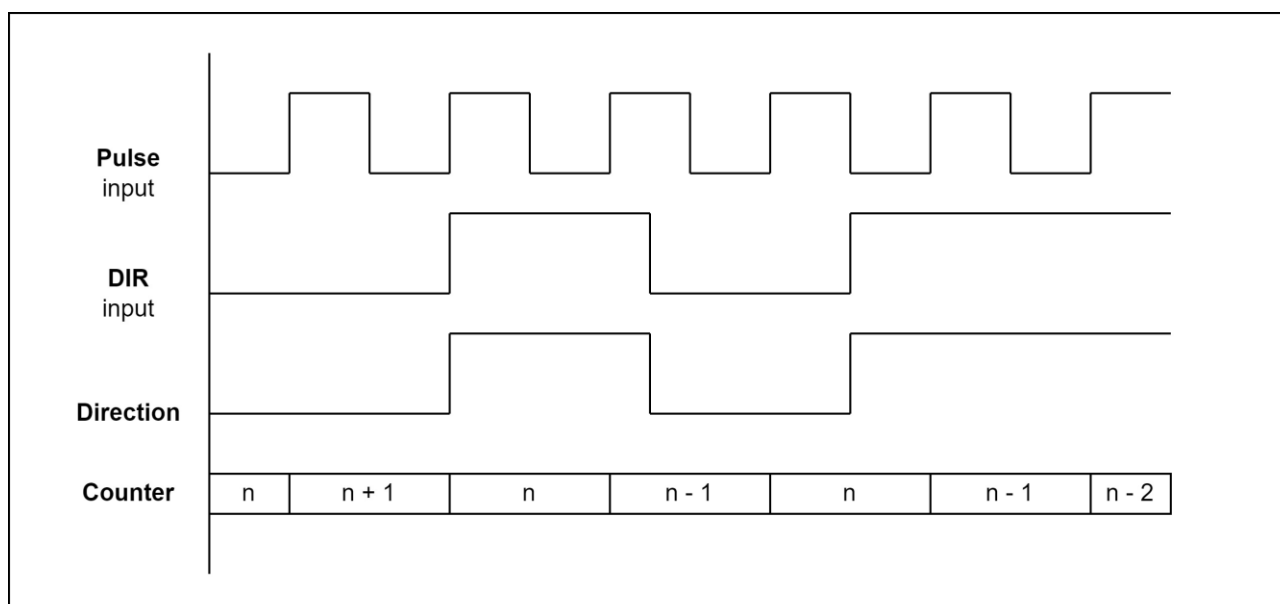
### Description

Configure the encoder mode of the specified encoder on the EtherCAT SubDevice. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

This EtherCAT SubDevice supports a total of six encoder modes, as follows:

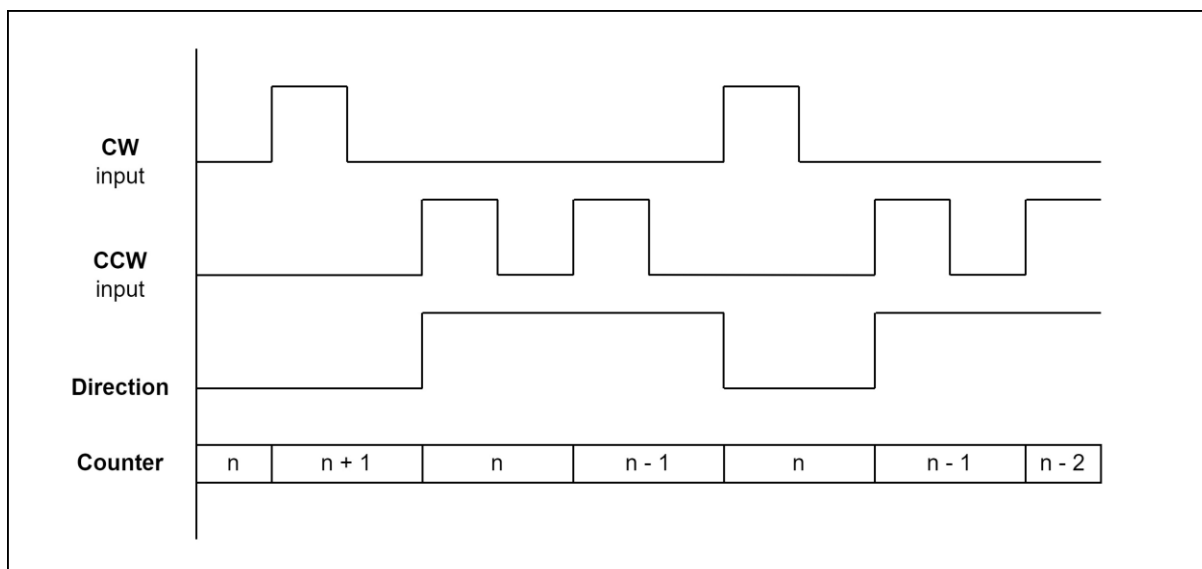
- **Pulse/DIR (1-Pulse mode)**

This mode is also known as 1-Pulse mode and has two input pins for accepting signals from the encoder: the Pulse input and the DIR input. The Pulse input is used to count the number of pulses from the encoder, and the DIR input is used to indicate the current counting direction of the encoder counter. In this mode, the counter only counts on the rising edge of the Pulse input.



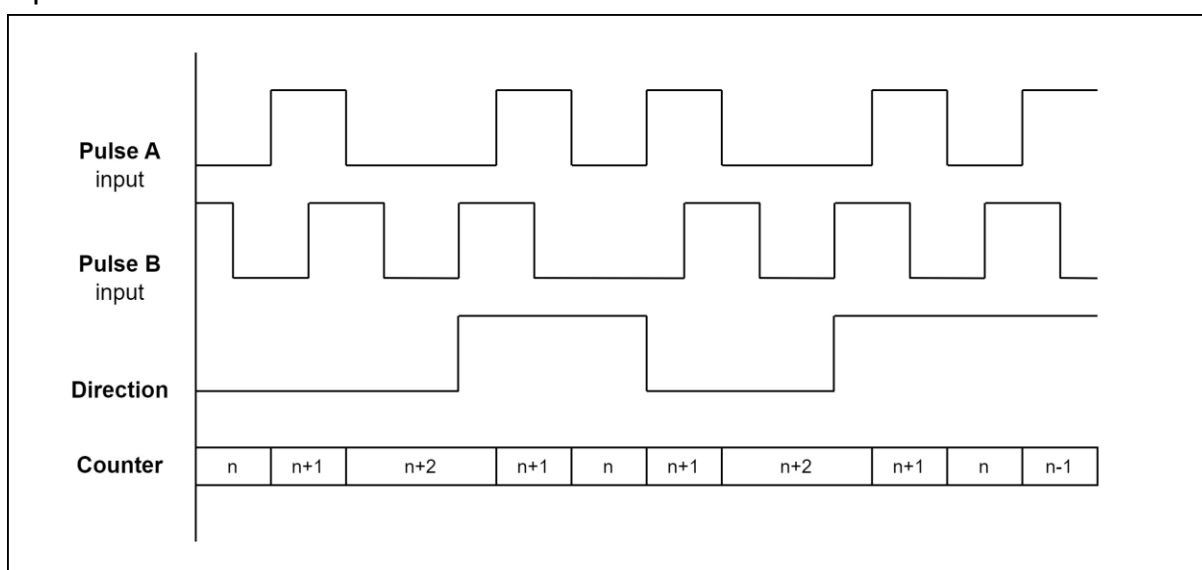
- **CW/CCW (2-Pulse mode)**

This mode is also known as 2-Pulse mode. It has two input pins for accepting signals from the encoder: the CW input and the CCW input. The pulses from the CW input represent the number of pulses counted in the forward direction by the encoder counter, while the pulses from the CCW input represent the number of pulses counted in the reverse direction by the encoder counter. In this mode, the counter only counts on the rising edge of the CW and CCW inputs.



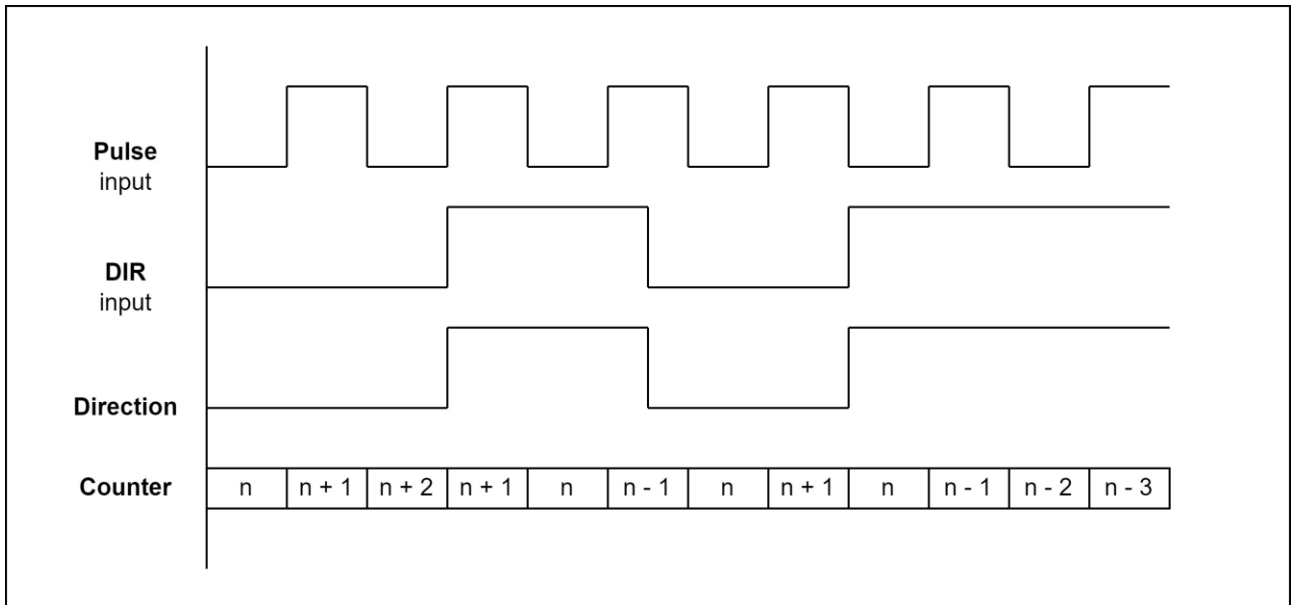
- **Pulse A/B (QEI mode)**

This mode is also known as QEI (Quadrature Encoder Interface) mode. It has two input pins for accepting signals from the encoder: the Pulse A input and the Pulse B input. The Pulse A and Pulse B inputs have a unique relationship. If Pulse A leads Pulse B, the counting direction is deemed forward. If Pulse A lags Pulse B, the counting direction is deemed reverse. In this mode, the counter only counts on the rising and falling edges of the Pulse A input.



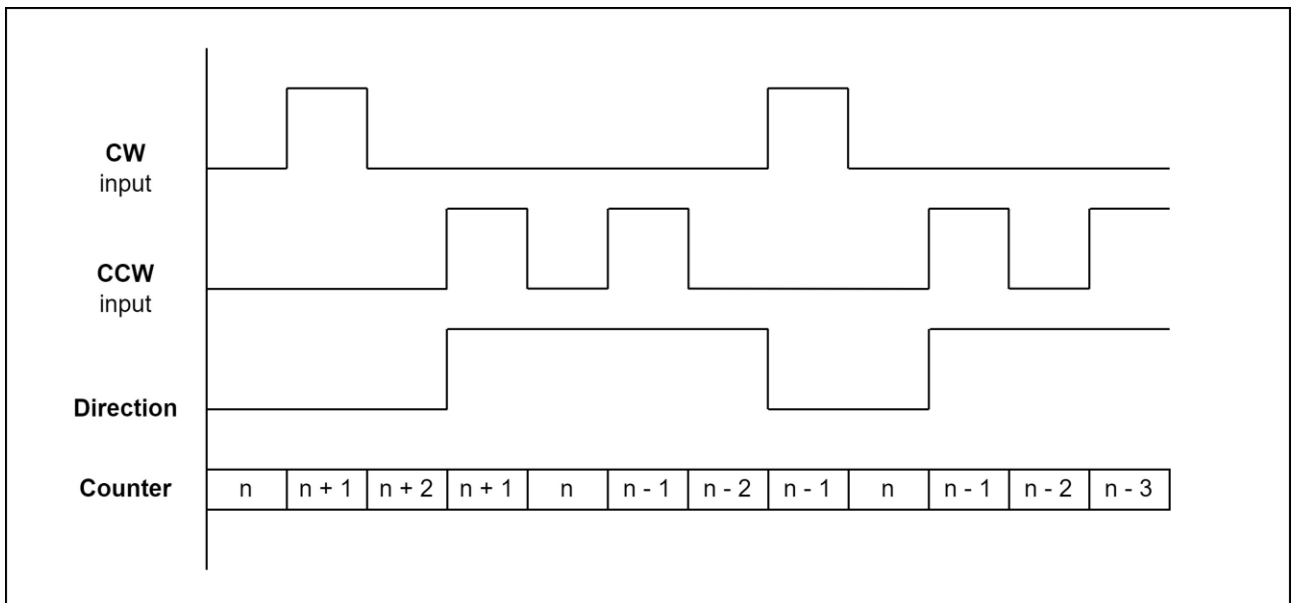
- **Pulse/DIR x2**

This mode is similar to **Pulse/DIR** mode, with the only difference being that the counter in this mode counts on both the rising and falling edges of the Pulse input.



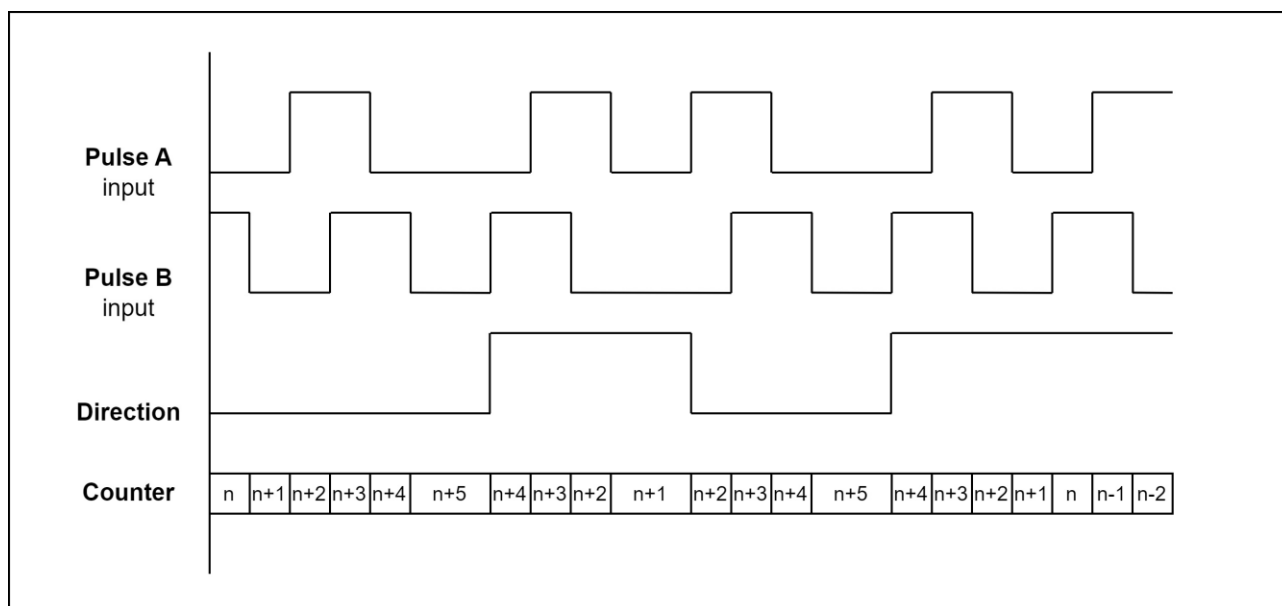
- **CW/CCW x2**

This mode is similar to **CW/CCW** mode, with the only difference being that the counter in this mode counts on both the rising and falling edges of the CW and CCW inputs.



- **Pulse A/B x2**

This mode is similar to **Pulse A/B** mode, with the only difference being that the counter in this mode counts on both the rising and falling edges of the Pulse A and Pulse B inputs.



### Syntax

```
int configEncoderMode(int encoder, uint8_t mode);
```

### Parameters

- **[in] encoder**

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by [configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- **[in] mode**

The encoder mode to be configured. Users can select from six encoder modes, as detailed in the table below. For input values not listed in the table, the default configuration will be ECAT\_ENCODER\_MODE\_AB\_PHASE\_x2.

Definition	Value	Description
ECAT_ENCODER_MODE_STEP_DIR	0	Please see Pulse/DIR mode.
ECAT_ENCODER_MODE_CWCCW	1	Please see CW/CCW mode.
ECAT_ENCODER_MODE_AB_PHASE	2	Please see Pulse A/B mode.
ECAT_ENCODER_MODE_STEP_DIR_x2	5	Please see Pulse/DIR x2 mode.
ECAT_ENCODER_MODE_CWCCW_x2	6	Please see CW/CCW x2 mode.
ECAT_ENCODER_MODE_AB_PHASE_x2	7	Please see Pulse A/B x2 mode.

## Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

## Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

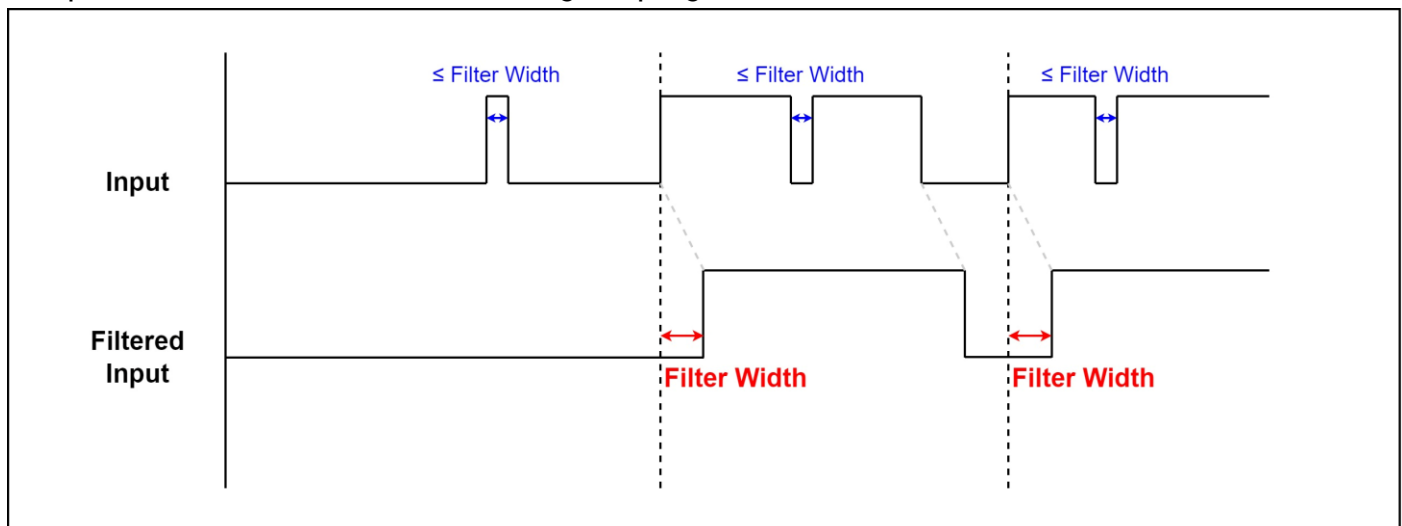
  device.configEncoderMode(ECAT_ENCODER_1, ECAT_ENCODER_MODE_AB_PHASE_x2);
  device.configEncoderMode(ECAT_ENCODER_2, ECAT_ENCODER_MODE_AB_PHASE_x2);
  device.configEncoderMode(ECAT_ENCODER_3, ECAT_ENCODER_MODE_AB_PHASE_x2);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configEncoderDigitalFilter()

### Description

Configure the digital filter of the specified encoder on the EtherCAT SubDevice. As shown in the figure, pulses with a width smaller than the filter bandwidth are filtered out, and pulses that are not filtered out are delayed by the time of the filter bandwidth. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.



### Syntax

```
int configEncoderDigitalFilter(int encoder, uint32_t width);
```

### Parameters

- [in] encoder

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by [configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- [in] width

The digital filter of the encoder to be configured is in units of 10 nanoseconds. If this value is 100, it means the filter width is 1000 nanoseconds.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

## Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

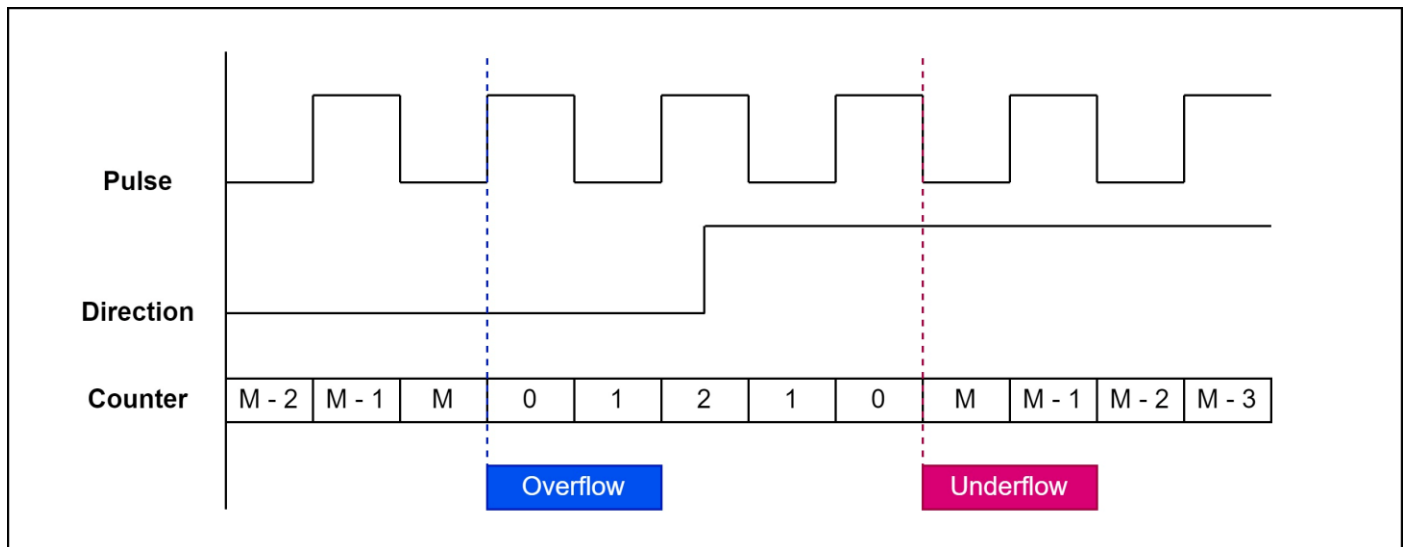
  device.configEncoderDigitalFilter(ECAT_ENCODER_1, 100);
  device.configEncoderDigitalFilter(ECAT_ENCODER_2, 100);
  device.configEncoderDigitalFilter(ECAT_ENCODER_3, 100);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configEncoderRange()

### Description

Configure the maximum value of the counter for the specified encoder on the EtherCAT SubDevice. As shown in the figure, upon reaching its maximum value (M), the counter overflows, resets to zero, and triggers an [Overflow](#) event. Conversely, when the counter decrements to zero, it underflows, resets to the maximum value, and triggers an [Underflow](#) event. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.



### Syntax

```
int configEncoderRange(int encoder, uint32_t value);
```

### Parameters

- `[in] encoder`

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by [configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- `[in] value`

The maximum value of the encoder counter to be configured.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

## Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

## Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderRange(ECAT_ENCODER_1, 8000);
  device.configEncoderRange(ECAT_ENCODER_2, 8000);
  device.configEncoderRange(ECAT_ENCODER_3, 8000);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configEncoderInputPolarity()

### Description

Configure the polarity of the input pins for the specified encoder on the EtherCAT SubDevice. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.

### Syntax

```
int configEncoderInputPolarity(int encoder, bool pin_a, bool pin_b, bool pin_z);
```

### Parameters

- `[in] encoder`

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by [configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- `[in] pin_a`

The polarity of input pin A for the specified encoder to be configured.

- `true`: High voltage represents logic 1, while low voltage represents logic 0.
- `false`: High voltage represents logic 0, while low voltage represents logic 1.

- `[in] pin_b`

The polarity of input pin B for the specified encoder to be configured. The explanation for the input value is the same as `pin_a`.

- `[in] pin_z`

The polarity of input pin Z for the specified encoder to be configured. The explanation for the input value is the same as `pin_a`.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
```

```
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

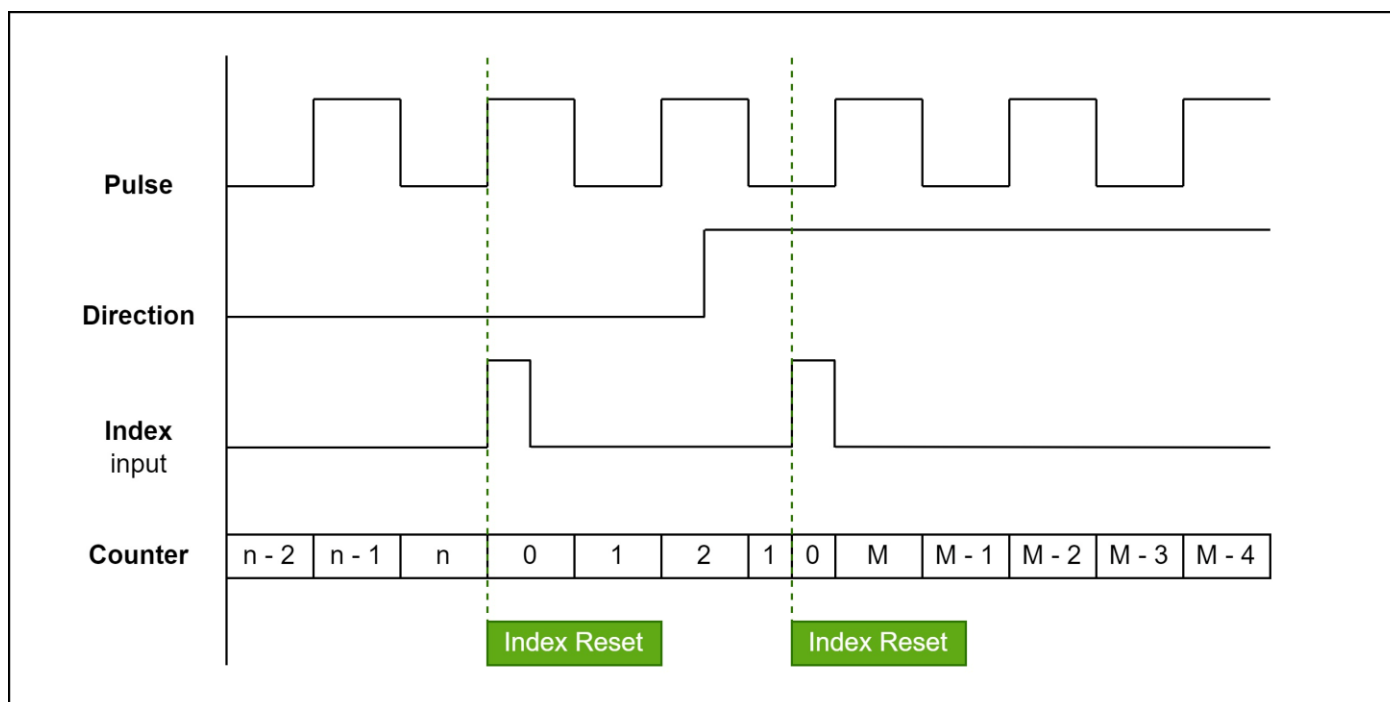
  device.configEncoderInputPolarity(ECAT_ENCODER_1, true, true, true);
  device.configEncoderInputPolarity(ECAT_ENCODER_2, true, true, true);
  device.configEncoderInputPolarity(ECAT_ENCODER_3, true, true, true);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## configEncoderIndexReset()

### Description

Enable or disable the index signal reset counter function for the specified encoder on the EtherCAT SubDevice. As shown in the figure, upon enabling this function, the counter will be reset to zero and the [Index Reset](#) event will be triggered when the index signal (signal Z) is detected. This parameter is written to the EEPROM of the EtherCAT SubDevice and loaded during startup, so users do not need to configure this parameter each time before running the program.



### Syntax

```
int configEncoderIndexReset(int encoder, bool enable);
```

### Parameters

- [in] encoder

The specified encoder number:

Definition	Value	Description
ECAT_ENCODER_1	0x01	Encoder 1 on the EtherCAT SubDevice.
ECAT_ENCODER_2	0x02	Encoder 2 on the EtherCAT SubDevice.
ECAT_ENCODER_3	0x03	Encoder 3 on the EtherCAT SubDevice.
ECAT_ENCODER_X	0x11	The encoder which is mapped to the X-axis.
ECAT_ENCODER_Y	0x12	The encoder which is mapped to the Y-axis.
ECAT_ENCODER_Z	0x13	The encoder which is mapped to the Z-axis.

The mapping of the encoders to the mechanical axes are determined by [configMachineAxisMapping\(\)](#) and [configMachinePositionFeedbackSource\(\)](#).

- `[in] enable`  
A Boolean value that specifies whether to enable or disable the index signal reset function for the specified encoder.
  - `true`: The index signal reset function will be enabled.
  - `false`: The index signal reset function will be disabled.

### Return Value

Return an [error code](#). If the returned value is zero, it indicates a successful execution of this function.

### Comment

This function must be called after a successful execution of [EthercatMaster::begin\(\)](#). This function is blocking and cannot be called in callback functions.

### Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_QECR11MP3S device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.configEncoderIndexReset(ECAT_ENCODER_1, false);
  device.configEncoderIndexReset(ECAT_ENCODER_2, false);
  device.configEncoderIndexReset(ECAT_ENCODER_3, false);
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

# Ch. 3

## Examples

## 3.1 SubDevice Information Examples

SubDevice Information Examples.

### 3.1.1 Example 1: Using EthercatMaster class

Show SubDevice information using EthercatMaster class.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;

void setup() {
  Serial.begin(115200);
  master.begin();

  for (int i = 0; i < master.getSlaveCount(); i++) {
    Serial.print("SubDevice");
    if (i < 10) Serial.print(" ");
    Serial.print(i);

    Serial.print(" VID:");
    Serial.print(master.getVendorID(i), HEX);
    Serial.print("h, PID:");
    Serial.print(master.getProductCode(i), HEX);
    Serial.print("h, Rev:");
    Serial.print(master.getRevisionNumber(i), HEX);
    Serial.print("h, Ser:");
    Serial.print(master.getSerialNumber(i), HEX);
    Serial.print("h, Alias:");
    Serial.print(master.getAliasAddress(i));
    Serial.println();
  }
}

void loop() {
}
```

## 3.1.2 Example 2: Using EthercatDevice\_Generic class

Show SubDevice information using EthercatDevice\_Generic class.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  char name[256];

  master.begin();

  for (int i = 0; i < master.getSlaveCount(); i++) {
    device.attach(i, master);

    Serial.print("SubDevice");
    if (i < 10) Serial.print(" ");
    Serial.println(i);

    Serial.print("          Name: ");
    Serial.println(device.getDeviceName(name, 256));

    Serial.print("          Vendor ID: 0x");
    Serial.println(device.getVendorID(), HEX);

    Serial.print("          Product Code: 0x");
    Serial.println(device.getProductCode(), HEX);

    Serial.print("          Revision Number: 0x");
    Serial.println(device.getRevisionNumber(), HEX);

    Serial.print("          Serial Number: 0x");
    Serial.println(device.getSerialNumber(), HEX);

    Serial.print("          Alias Address: ");
    Serial.println(device.getAliasAddress());
```

```
Serial.print("    Mailbox Protocol: 0x");
Serial.println(device.getMailboxProtocol(), HEX);

Serial.print("        CoE Details: 0x");
Serial.println(device.getCoEDetails(), HEX);

Serial.print("        FoE Details: 0x");
Serial.println(device.getFoEDetails(), HEX);

Serial.print("        EoE Details: 0x");
Serial.println(device.getEoEDetails(), HEX);

Serial.print("        SoE Channels: ");
Serial.println(device.getSoEChannels());

Serial.print("        DC Supported: ");
Serial.println(device.isSupportDC());
}
}

void loop() {
}
```

### 3.1.3 Example 3: Using EthercatDevice\_CiA402 class

Show SubDevice information using EthercatDevice\_CiA402 class.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 device;

void setup() {
  Serial.begin(115200);

  char name[256];

  master.begin();

  for (int i = 0; i < master.getSlaveCount(); i++) {
    if (device.attach(i, master) < 0) {
      continue;
    }

    Serial.print("SubDevice");
    if (i < 10) Serial.print(" ");
    Serial.println(i);

    Serial.print("          Name: ");
    Serial.println(device.getDeviceName(name, 256));

    Serial.print("          Vendor ID: 0x");
    Serial.println(device.getVendorID(), HEX);

    Serial.print("          Product Code: 0x");
    Serial.println(device.getProductCode(), HEX);

    Serial.print("          Revision Number: 0x");
    Serial.println(device.getRevisionNumber(), HEX);

    Serial.print("          Serial Number: 0x");
    Serial.println(device.getSerialNumber(), HEX);
  }
}
```

```
Serial.print("      Alias Address: ");
Serial.println(device.getAliasAddress());

Serial.print("      Mailbox Protocol: 0x");
Serial.println(device.getMailboxProtocol(), HEX);

Serial.print("      CoE Details: 0x");
Serial.println(device.getCoEDetails(), HEX);

Serial.print("      FoE Details: 0x");
Serial.println(device.getFoEDetails(), HEX);

Serial.print("      EoE Details: 0x");
Serial.println(device.getEoEDetails(), HEX);

Serial.print("      SoE Channels: ");
Serial.println(device.getSoEChannels());

Serial.print("      DC Supported: ");
Serial.println(device.isSupportDC());
}
}

void loop() {
}
```

## 3.2 SDO Upload/Download Examples

SDO Upload/Download Examples.

### 3.2.1 Example 1: SDO Upload using `sdoUpload8()`

The usage of `sdoUpload16()`, `sdoUpload32()`, and `sdoUpload64()` is similar to `sdoUpload8()`, except for the difference in the return value type.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);

  Serial.print("1C12h.0 => ");
  Serial.println(device.sdoUpload8(0x1C12, 0x00));

  Serial.print("1C13h.0 => ");
  Serial.println(device.sdoUpload8(0x1C13, 0x00));
  // ...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 3.2.2 Example 2: SDO Upload using `sdoUpload()`

SDO Upload using [sdoUpload\(\)](#).

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint8_t value;

  master.begin();
  device.attach(0, master);

  if (device.sdoUpload(0x1C12, 0x00, &value, sizeof(value)) >= 0) {
    Serial.print("1C12h.0 => ");
    Serial.println(value);
  }

  if (device.sdoUpload(0x1C13, 0x00, &value, sizeof(value)) >= 0) {
    Serial.print("1C13h.0 => ");
    Serial.println(value);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

### 3.2.3 Example 3: SDO Upload using `sdoUpload()` with abort code

SDO Upload using `sdoUpload()` with abort code.

Initiate an SDO Upload command to read a value from a non-existent object, expecting an abort code of 0x06020000. For more information about abort codes, please refer to [SDO Abort Code](#).

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint32_t abortcode;
  uint8_t value;

  master.begin();
  device.attach(0, master);

  if (device.sdoUpload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
  ECAT_ERR_DEVICE_COE_ERROR) {
    Serial.print("Abort Code: 0x");
    Serial.println(abortcode, HEX);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

### 3.2.4 Example 4: SDO Download using sdoDownload8()

SDO Download using [sdoDownload8\(\)](#).

The usage of [sdoDownload16\(\)](#), [sdoDownload32\(\)](#), and [sdoDownload64\(\)](#) is similar to [sdoDownload8\(\)](#), except for the difference in the input parameter types.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);

  device.sdoDownload8(0x1C12, 0x00, 0);
  device.sdoDownload8(0x1C13, 0x00, 0);
  //...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

### 3.2.5 Example 5: SDO Download using sdoDownload()

SDO Download using [sdoDownload\(\)](#).

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  uint8_t value;

  master.begin();
  device.attach(0, master);

  value = 0;
  device.sdoDownload(0x1C12, 0x00, &value, sizeof(value));
  value = 0;
  device.sdoDownload(0x1C13, 0x00, &value, sizeof(value));
  //...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

### 3.2.6 Example 6: SDO Download using `sdoDownload()` with abort code

SDO Download using `sdoDownload()` with abort code.

Initiate an SDO Download command to write a value to a non-existent object, expecting an abort code of 0x06020000. For more information about abort codes, please refer to [SDO Abort Code](#).

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint32_t abortcode;
  uint8_t value = 0x01;

  master.begin();
  device.attach(0, master);

  if (device.sdoDownload(0xFFFF, 0xFF, &value, sizeof(value), &abortcode) ==
  ECAT_ERR_DEVICE_COE_ERROR) {
    char buf[20];
    sprintf(buf, "Abort Code: 0x%08lX", abortcode);
    Serial.println(buf);
  }
  //...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

### 3.2.7 Example 7: Print the PDO mapping configuration

Print the PDO mapping configuration.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  uint8_t assign_nr, mapping_nr;
  uint16_t mapping;
  uint32_t entry;

  master.begin();
  device.attach(0, master);

  // RxPDO Mapping List
  assign_nr = device.sdoUpload8(0x1C12, 0x00);
  for (int m = 0; m < assign_nr; m++) {
    mapping = device.sdoUpload16(0x1C12, m + 1);
    Serial.print(" RxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h");

    mapping_nr = device.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
      entry = device.sdoUpload32(mapping, n + 1);
      Serial.print("  ");
      char entryBuf[11]; // "XXXXXXXXXh" + null
      sprintf(entryBuf, "%08lXh", entry);
      Serial.println(entryBuf);
    }
  }

  // TxPDO Mapping List
```

```
assign_nr = device.sdoUpload8(0x1C13, 0x00);
for (int m = 0; m < assign_nr; m++) {
    mapping = device.sdoUpload16(0x1C13, m + 1);
    Serial.print(" TxPDO");
    Serial.print(m + 1);
    Serial.print(" (");
    Serial.print(mapping, HEX);
    Serial.println("h");

    mapping_nr = device.sdoUpload8(mapping, 0x00);
    for (int n = 0; n < mapping_nr; n++) {
        entry = device.sdoUpload32(mapping, n + 1);
        Serial.print(" ");
        char entryBuf[11];
        sprintf(entryBuf, "%08lXh", entry);
        Serial.println(entryBuf);
    }
}
//...
}

void loop() {
    // put your main code here, to run repeatedly:

}
```

### 3.2.8 Example 8: Change the PDO mapping configuration

Change the PDO mapping configuration.

Map the following objects to PDOs in a CiA 402 device that supports PDO mapping:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 607Ah: Target position
  - Object 60FFh: Target velocity
  - Object 6071h: Target torque
  - Object 6060h: Modes of operation
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 6064h: Position actual value
  - Object 606Ch: Velocity actual value
  - Object 6077h: Torque actual value
  - Object 6061h: Modes of operation display

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  // 1C12
  device.sdoDownload8(0x1C12, 0x00, 0);
  device.sdoDownload8(0x1601, 0x00, 0);
  device.sdoDownload32(0x1601, 0x01, 0x60400010);
  device.sdoDownload32(0x1601, 0x02, 0x607A0020);
  device.sdoDownload32(0x1601, 0x03, 0x60FF0020);
  device.sdoDownload32(0x1601, 0x04, 0x60710010);
  device.sdoDownload32(0x1601, 0x05, 0x60600008);
  device.sdoDownload8(0x1601, 0x00, 5);
  device.sdoDownload16(0x1C12, 0x1601, 1);
  device.sdoDownload8(0x1C12, 0x00, 1);
  // 1C13
  device.sdoDownload8(0x1C13, 0x00, 0);
  device.sdoDownload8(0x1A01, 0x00, 0);
  device.sdoDownload32(0x1A01, 0x01, 0x60410010);
```

```
device.sdoDownload32(0x1A01, 0x02, 0x60640020);
device.sdoDownload32(0x1A01, 0x03, 0x606C0020);
device.sdoDownload32(0x1A01, 0x04, 0x60770010);
device.sdoDownload32(0x1A01, 0x05, 0x60610008);
device.sdoDownload8(0x1A01, 0x00, 5);
device.sdoDownload16(0x1C13, 0x1A01, 1);
device.sdoDownload8(0x1C13, 0x00, 1);
//...
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

## 3.3 PDO Read/Write Examples

PDO Read/Write Examples.

### 3.3.1 Example 1: Read a bit data from Input PDO using pdoBitRead()

Read a bit from Input PDO using [pdoBitRead\(\)](#).

A 16-channel digital input EtherCAT SubDevice has 2-byte Input PDOs, with each bit corresponding to a digital input channel. The states of channels 0 and 9 will be printed with a frequency of 1 Hz.

B15	B14	B13	B12	B11	B10	<b>B9</b>	B8	B7	B6	B5	B4	B3	B2	B1	<b>B0</b>
-----	-----	-----	-----	-----	-----	-----------	----	----	----	----	----	----	----	----	-----------

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // Read and display the state of PDO bits
  Serial.print("Bit0 => ");
  Serial.print(device.pdoBitRead(0)); // Read PDO bit 0
  Serial.print(", Bit9 => ");
  Serial.println(device.pdoBitRead(9)); // Read PDO bit 9

  delay(1000);
}
```

### 3.3.2 Example 2: Read a byte data from Input PDO using pdoRead8()

Read data from Input PDO using [pdoRead8\(\)](#).

Same as example 1.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  // Read specific bits using pdoRead8 and print their values
  Serial.print("Bit0 => ");
  Serial.print((device.pdoRead8(0) >> 0) & 1);
  Serial.print(", Bit9 => ");
  Serial.println((device.pdoRead8(1) >> 1) & 1);

  delay(1000);
}
```

### 3.3.3 Example 3: Read data from Input PDO using pdoRead()

Read data from Input PDO using [pdoRead\(\)](#).

Same as example 1.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value;

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoRead(0, &value, sizeof(value));

  Serial.print("Bit0 => ");
  Serial.print((value >> 0) & 1);
  Serial.print(", Bit9 => ");
  Serial.println((value >> 9) & 1);

  delay(1000);
}
```

### 3.3.4 Example 4: Write a bit data to Output PDO using pdoBitWrite()

Write a bit to Output PDO using [pdoBitWrite\(\)](#).

A 16-channel digital output EtherCAT SubDevice has 2-byte Output PDOs, with each bit corresponding to a digital output channel. Channels 0 and 9 will be toggled at a frequency of 1 Hz.

B15	B14	B13	B12	B11	B10	<b>B9</b>	B8	B7	B6	B5	B4	B3	B2	B1	<b>B0</b>
-----	-----	-----	-----	-----	-----	-----------	----	----	----	----	----	----	----	----	-----------

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoBitWrite(0, 1);
  device.pdoBitWrite(9, 1);
  delay(1000);

  device.pdoBitWrite(0, 0);
  device.pdoBitWrite(9, 0);
  delay(1000);
}
```

### 3.3.5 Example 5: Write a byte data to Output PDO using `pdoWrite8()`

Write data to Output PDO using `pdoWrite8()`.

Same as example 4.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  device.pdoWrite8(0, 0x01);
  device.pdoWrite8(1, 0x02);
  delay(1000);

  device.pdoWrite8(0, 0x00);
  device.pdoWrite8(1, 0x00);
  delay(1000);
}
```

### 3.3.6 Example 6: Write data to Output PDO using `pdoWrite()`

Write data to Output PDO using `pdoWrite()`.

Same as examples 4.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint16_t value;

void setup() {
  master.begin();
  device.attach(0, master);
  master.start();
}

void loop() {
  value = 0x0201;
  device.pdoWrite(0, &value, sizeof(value));
  delay(1000);

  value = 0x0000;
  device.pdoWrite(0, &value, sizeof(value));
  delay(1000);
}
```

## 3.4 Callback Examples

Callback Examples.

### 3.4.1 Example 1: Cyclic callback

A 16-channel digital output EtherCAT SubDevice has 2-byte Output PDOs, with each bit corresponding to a digital output channel. Channels 0 and 9 will be toggled at a frequency of 1 Hz.

B15	B14	B13	B12	B11	B10	<b>B9</b>	B8	B7	B6	B5	B4	B3	B2	B1	<b>B0</b>
-----	-----	-----	-----	-----	-----	-----------	----	----	----	----	----	----	----	----	-----------

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

int toggle = 0;
int cycle_count = 0;

void MyCyclicCallback()
{
  if (++cycle_count < 1000)
    return;

  cycle_count = 0;

  toggle = !toggle;
  device.pdoBitWrite(0, toggle);
  device.pdoBitWrite(9, toggle);
}

void setup() {
  Serial.begin(115200);

  master.begin();
  device.attach(0, master);
  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000); // 1ms cycle
```

```
}  
  
void loop() {  
  // put your main code here, to run repeatedly:  
  
}
```

### 3.4.2 Example 2: Cyclic callback with FPU-enabled

Cyclic callback with FPU-enabled.

If the callback function requires floating-point arithmetic, please use FPU-enabled callback function. For more detailed information, please refer to the function description below.

- [attachCyclicCallback\(\)](#)
- [attachErrorCallback\(\)](#)
- [attachEventCallback\(\)](#)

Here is the example code:

```
#include "Ethercat.h"

#define GRAVITY (9.80665)

EthercatMaster master;
EthercatDevice_Generic device;

int toggle = 0;
int cycle_count = 0;
double S = 0.0, T = 0.0;

double s = 0.0, t = 0.0;

void MyCyclicCallback()
{
  S = (GRAVITY * T * T) / 2.0;
  T += 0.001;

  if (++cycle_count < 1000)
    return;
  cycle_count = 0;

  toggle = !toggle;
  device.pdoBitWrite(0, toggle);
  device.pdoBitWrite(9, toggle);
}

void setup() {
  Serial.begin(115200);
```

```
master.begin();
device.attach(0, master);
master.attachCyclicCallback(MyCyclicCallback, true);
master.start(1000000);
}

void loop() {
  s = (GRAVITY * t * t) / 2.0;
  t += 1.0;

  Serial.print("S = ");
  Serial.println(s, 6);

  delay(1000);
}
```

### 3.4.3 Example 3: Error callback

Print the count of each type of error once per second.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

// Error counters
int wkc_single_fault_cnt = 0;
int wkc_multiple_faults_cnt = 0;
int single_lost_frame_cnt = 0;
int multiple_lost_frames_cnt = 0;
int cable_broken_cnt = 0;
int wait_ack_timeout_cnt = 0;

// Error callback function
void myErroCallback(uint32_t errorcode) {
    switch (errorcode) {
        case ECAT_ERR_WKC_SINGLE_FAULT:
            wkc_single_fault_cnt++;
            break;
        case ECAT_ERR_WKC_MULTIPLE_FAULTS:
            wkc_multiple_faults_cnt++;
            break;
        case ECAT_ERR_SINGLE_LOST_FRAME:
            single_lost_frame_cnt++;
            break;
        case ECAT_ERR_MULTIPLE_LOST_FRAMES:
            multiple_lost_frames_cnt++;
            break;
        case ECAT_ERR_CABLE_BROKEN:
            cable_broken_cnt++;
            break;
        case ECAT_ERR_WAIT_ACK_TIMEOUT:
            wait_ack_timeout_cnt++;
            break;
    }
}
```

```
void setup() {
  Serial.begin(115200);

  master.attachErrorCallback(myErroCallback);
  master.begin();
  master.start();
}

void loop() {

  Serial.print("ECAT_ERR_WKC_SINGLE_FAULT      = ");
  Serial.println(wkc_single_fault_cnt);
  Serial.print("ECAT_ERR_WKC_MULTIPLE_FAULTS  = ");
  Serial.println(wkc_multiple_faults_cnt);
  Serial.print("ECAT_ERR_SINGLE_LOST_FRAME    = ");
  Serial.println(single_lost_frame_cnt);
  Serial.print("ECAT_ERR_MULTIPLE_LOST_FRAMES = ");
  Serial.println(multiple_lost_frames_cnt);
  Serial.print("ECAT_ERR_CABLE_BROKEN          = ");
  Serial.println(cable_broken_cnt);
  Serial.print("ECAT_ERR_WAIT_ACK_TIMEOUT      = ");
  Serial.println(wait_ack_timeout_cnt);

  delay(1000);
}
```

### 3.4.4 Example 4: Event callback

Print the count of each type of event once per second.

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

// Event counters
int state_changed_cnt = 0;
int cable_reconnected_cnt = 0;

// Event callback function
void myEventCallback(uint32_t eventcode) {
  switch (eventcode) {
    case ECAT_EVT_STATE_CHANGED:
      state_changed_cnt++;
      break;
    case ECAT_EVT_CABLE_RECONNECTED:
      cable_reconnected_cnt++;
      break;
  }
}

void setup() {
  Serial.begin(115200);

  master.attachEventCallback(myEventCallback);
  master.begin();
  master.start();
}

void loop() {
  Serial.print("ECAT_EVT_STATE_CHANGED    = ");
  Serial.println(state_changed_cnt);
  Serial.print("ECAT_EVT_CABLE_RECONNECTED = ");
  Serial.println(cable_reconnected_cnt);
  delay(1000);
}
```

## 3.5 DC Examples

DC Examples.

### 3.5.1 Example 1: Enable DC synchronization

Enable DC synchronization.

Implementing position control on a CiA 402 EtherCAT SubDevice using the EthercatDevice\_Generic class. The CiA 402 control mode is set to cyclic synchronous position mode, and DC synchronization is enabled for precise timing.

The default PDO mapping is as follows:

- **Output PDO (RxPDO)**

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Controlword		Target Position			

- **Input PDO (TxPDO)**

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
Statusword		Position Actual Value			

Here is the example code:

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_Generic device;

uint32_t position = 0;

// Cyclic callback function, executed every 1ms
void MyCyclicCallback() {
    // Check if the device is in "Operation Enable" state (0x27)
    if ((device.pdoRead8(0) & 0x6F) != 0x27)
        return;

    // Increment target position by 1000 and write it to PDO
    device.pdoWrite32(2, position += 1000);
}
```

```
void setup() {
  Serial.begin(115200);
  while (!Serial); // Wait for Serial to be ready

  master.begin();

  device.attach(0, master);           // Attach SubDevice (device ID 0)
  device.setDc(1000000);              // Enable Distributed Clock (1ms cycle)
  device.sdoDownload8(0x6060, 0x00, 8); // Set mode of operation to CSP (mode
8)

  // Register the cyclic callback function
  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000, ECAT_SYNC);   // Start EtherCAT master with 1ms sync
cycle

  // Read current position and set as initial target position
  position = device.pdoRead32(2);
  device.pdoWrite32(2, position);

  // Transition through CiA 402 state machine to reach "Operation Enable"
  device.pdoWrite8(0, 0x80); // Shutdown
  delay(1000);
  device.pdoWrite8(0, 0x06); // Switch On
  delay(1000);
  device.pdoWrite8(0, 0x07); // Enable Voltage
  delay(1000);
  device.pdoWrite8(0, 0x0F); // Operation Enable
  delay(1000);
}

void loop() {
  // put your main code here, to run repeatedly:
}
```



# Appendix

## A.1 Error List

For most functions, a returned value less than zero indicates an error, and the value represents an error code.

If there is an error code, you can find the error cause and corrective actions below.

Definition	Code
<a href="#">ECAT_SUCCESS</a>	0
<a href="#">ECAT_ERR_MODULE_INIT_FAIL</a>	-100
<a href="#">ECAT_ERR_MODULE_GET_VERSION_FAIL</a>	-101
<a href="#">ECAT_ERR_MODULE_VERSION_MISMATCH</a>	-102
<a href="#">ECAT_ERR_MODULE_GENERIC_TRANSFER_INIT_FAIL</a>	-103
<a href="#">ECAT_ERR_MASTER_DOWNLOAD_SETTINGS_FAIL</a>	-200
<a href="#">ECAT_ERR_MASTER_SET_DEVICE_SETTINGS_FAIL</a>	-201
<a href="#">ECAT_ERR_MASTER_GET_GROUP_INFO_FAIL</a>	-202
<a href="#">ECAT_ERR_MASTER_GET_MASTER_INFO_FAIL</a>	-203
<a href="#">ECAT_ERR_MASTER_GET_DEVICE_INFO_FAIL</a>	-204
<a href="#">ECAT_ERR_MASTER_SET_GROUP_SETTINGS_FAIL</a>	-205
<a href="#">ECAT_ERR_MASTER_MAPPING_INIT_FAIL</a>	-206
<a href="#">ECAT_ERR_MASTER_INTERRUPT_INIT_FAIL</a>	-207
<a href="#">ECAT_ERR_MASTER_ACTIVE_FAIL</a>	-208
<a href="#">ECAT_ERR_MASTER_ENI_INITCMDS_FAIL</a>	-209
<a href="#">ECAT_ERR_MASTER_NO_DEVICE</a>	-210
<a href="#">ECAT_ERR_MASTER_ACYCLIC_INIT_FAIL</a>	-300
<a href="#">ECAT_ERR_MASTER_ACYCLIC_REQUEST_FAIL</a>	-301
<a href="#">ECAT_ERR_MASTER_ACYCLIC_BUSY</a>	-302
<a href="#">ECAT_ERR_MASTER_ACYCLIC_TIMEOUT</a>	-303
<a href="#">ECAT_ERR_MASTER_ACYCLIC_ERROR</a>	-304
<a href="#">ECAT_ERR_MASTER_ACYCLIC_WRONG_STATUS</a>	-405
<a href="#">ECAT_ERR_MASTER_GENERIC_SEND_FAIL</a>	-400
<a href="#">ECAT_ERR_MASTER_GENERIC_RECV_FAIL</a>	-401
<a href="#">ECAT_ERR_MASTER_NOT_BEGIN</a>	-1000
<a href="#">ECAT_ERR_MASTER_WRONG_BUFFER_SIZE</a>	-1001
<a href="#">ECAT_ERR_MASTER_REDUNDANCY_NO_DC</a>	-1002
<a href="#">ECAT_ERR_MASTER_MEMORY_ALLOCATION_FAIL</a>	-1003
<a href="#">ECAT_ERR_MASTER_OSLAYER_INIT_FAIL</a>	-1004
<a href="#">ECAT_ERR_MASTER_NIC_INIT_FAIL</a>	-1005
<a href="#">ECAT_ERR_MASTER_BASE_INIT_FAIL</a>	-1006
<a href="#">ECAT_ERR_MASTER_CIA402_INIT_FAIL</a>	-1007

<a href="#">ECAT_ERR_MASTER_SETUP_PDO_FAIL</a>	-1008
<a href="#">ECAT_ERR_MASTER_SCAN_NETWORK_FAIL</a>	-1009
<a href="#">ECAT_ERR_MASTER_START_MASTER_FAIL</a>	-1010
<a href="#">ECAT_ERR_MASTER_CYCLETIME_TOO_SMALL</a>	-1011
<a href="#">ECAT_ERR_MASTER_DUMP_OUTPUT_PDO_FAIL</a>	-1012
<a href="#">ECAT_ERR_MASTER_CONFIG_DEVICE_FAIL</a>	-1013
<a href="#">ECAT_ERR_MASTER_CONFIG_MAPPING_FAIL</a>	-1014
<a href="#">ECAT_ERR_MASTER_WAIT_BUS_SYNC_TIMEOUT</a>	-1015
<a href="#">ECAT_ERR_MASTER_WAIT_MASTER_SYNC_TIMEOUT</a>	-1016
<a href="#">ECAT_ERR_MASTER_CYCLIC_START_FAIL</a>	-1017
<a href="#">ECAT_ERR_MASTER_WRONG_BUFFER_POINTER</a>	-1018
<a href="#">ECAT_ERR_MASTER_ENI_INIT_FAIL</a>	-1050
<a href="#">ECAT_ERR_MASTER_ENI_MISMATCH</a>	-1051
<a href="#">ECAT_ERR_MASTER_STOPPED</a>	-1100
<a href="#">ECAT_ERR_MASTER_STARTED</a>	-1101
<a href="#">ECAT_ERR_MASTER_NOT_IN_PREOP</a>	-1102
<a href="#">ECAT_ERR_MASTER_NOT_IN_SAFEOP</a>	-1103
<a href="#">ECAT_ERR_MASTER_NOT_IN_OP</a>	-1104
<a href="#">ECAT_ERR_MASTER_I1_TRANSITION_FAIL</a>	-1200
<a href="#">ECAT_ERR_MASTER_IP_TRANSITION_FAIL</a>	-1201
<a href="#">ECAT_ERR_MASTER_PS_TRANSITION_FAIL</a>	-1202
<a href="#">ECAT_ERR_MASTER_SO_TRANSITION_FAIL</a>	-1203
<a href="#">ECAT_ERR_DEVICE_NOT_EXIST</a>	-2000
<a href="#">ECAT_ERR_DEVICE_NOT_ATTACH</a>	-2001
<a href="#">ECAT_ERR_DEVICE_NO_MAILBOX</a>	-2002
<a href="#">ECAT_ERR_DEVICE_NO_DC</a>	-2003
<a href="#">ECAT_ERR_DEVICE_WRONG_INPUT</a>	-2004
<a href="#">ECAT_ERR_DEVICE_MEMORY_ALLOCATION_FAIL</a>	-2005
<a href="#">ECAT_ERR_DEVICE_VENDOR_ID_MISMATCH</a>	-2006
<a href="#">ECAT_ERR_DEVICE_PRODUCT_CODE_MISMATCH</a>	-2007
<a href="#">ECAT_ERR_DEVICE_NO_SUCH_FUNCTION</a>	-2008
<a href="#">ECAT_ERR_DEVICE_FUNCTION_NOT_INIT</a>	-2009
<a href="#">ECAT_ERR_DEVICE_BUSY</a>	-2010
<a href="#">ECAT_ERR_DEVICE_TIMEOUT</a>	-2011
<a href="#">ECAT_ERR_DEVICE_NO_DATA</a>	-2012
<a href="#">ECAT_ERR_DEVICE_SII_READ_FAIL</a>	-2100
<a href="#">ECAT_ERR_DEVICE_SII_WRITE_FAIL</a>	-2101
<a href="#">ECAT_ERR_DEVICE_PDO_NOT_EXIST</a>	-2200
<a href="#">ECAT_ERR_DEVICE_PDO_OUT_OF_RANGE</a>	-2201
<a href="#">ECAT_ERR_DEVICE_FOE_NOT_SUPPORT</a>	-2300
<a href="#">ECAT_ERR_DEVICE_FOE_REQUEST_FAIL</a>	-2310

<a href="#"><u>ECAT_ERR_DEVICE_FOE_TIMEOUT</u></a>	-2311
<a href="#"><u>ECAT_ERR_DEVICE_FOE_ERROR</u></a>	-2312
<a href="#"><u>ECAT_ERR_DEVICE_FOE_BUFFER_TOO_SMALL</u></a>	-2313
<a href="#"><u>ECAT_ERR_DEVICE_FOE_READ_FAIL</u></a>	-2314
<a href="#"><u>ECAT_ERR_DEVICE_FOE_WRITE_FAIL</u></a>	-2315
<a href="#"><u>ECAT_ERR_DEVICE_COE_SDO_NOT_SUPPORT</u></a>	-2400
<a href="#"><u>ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT</u></a>	-2401
<a href="#"><u>ECAT_ERR_DEVICE_COE_BUSY</u></a>	-2410
<a href="#"><u>ECAT_ERR_DEVICE_COE_REQUEST_FAIL</u></a>	-2411
<a href="#"><u>ECAT_ERR_DEVICE_COE_TIMEOUT</u></a>	-2412
<a href="#"><u>ECAT_ERR_DEVICE_COE_ERROR</u></a>	-2413
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_NOT_EXIST</u></a>	-2500
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_ADD_FAIL</u></a>	-2501
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_TYPE_MISMATCH</u></a>	-2502
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_NO_MODE_SUPPORT</u></a>	-2503
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_WRONG_MODE</u></a>	-2504
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_MODE_NOT_SUPPORT</u></a>	-2505
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_CHANGE_WRONG_STATE</u></a>	-2506
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_WRITE_OBJECT_FAIL</u></a>	-2507
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE</u></a>	-2580
<a href="#"><u>ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE_SOURCE</u></a>	-2581
<a href="#"><u>ECAT_ERR_DEVICE_EOE_NOT_SUPPORT</u></a>	-2600
<a href="#"><u>ECAT_ERR_DEVICE_EOE_NO_SUCH_PORT</u></a>	-2601
<a href="#"><u>ECAT_ERR_DEVICE_EOE_TOO_MUCH_CONTENT</u></a>	-2602
<a href="#"><u>ECAT_ERR_DEVICE_EOE_BUSY</u></a>	-2610
<a href="#"><u>ECAT_ERR_DEVICE_EOE_REQUEST_FAIL</u></a>	-2611
<a href="#"><u>ECAT_ERR_DEVICE_EOE_TIMEOUT</u></a>	-2612
<a href="#"><u>ECAT_ERR_GROUP_WRONG_INPUT</u></a>	-3000
<a href="#"><u>ECAT_ERR_GROUP_NOT_ATTACH</u></a>	-3001

## A.2 Error Description and Corrective Actions

0: ECAT\_SUCCESS

### **Description**

Function calls successful.

### **Corrective Actions**

Nothing to do.

-100: ECAT\_ERR\_MODULE\_INIT\_FAIL

### **Description**

EtherCAT firmware initialization error.

### **Corrective Actions**

Please contact the manufacturer.

-101: ECAT\_ERR\_MODULE\_GET\_VERSION\_FAIL

### **Description**

The command to obtain the EtherCAT firmware version encountered an error.

### **Corrective Actions**

Please contact the manufacturer.

-102: ECAT\_ERR\_MODULE\_VERSION\_MISMATCH

### **Description**

The EtherCAT firmware version does not match the EtherCAT library version.

### **Corrective Actions**

Please update the EtherCAT firmware. If this issue persists, please contact the manufacturer.

## -103: ECAT\_ERR\_MODULE\_GENERIC\_TRANSFER\_INIT\_FAIL

### **Description**

General transfer interface initialization between dual systems failed.

### **Corrective Actions**

Please contact the manufacturer.

## -200: ECAT\_ERR\_MASTER\_DOWNLOAD\_SETTINGS\_FAIL

### **Description**

The command to set the configuration of the EtherCAT master encountered an error.

### **Corrective Actions**

Please contact the manufacturer.

## -201: ECAT\_ERR\_MASTER\_SET\_DEVICE\_SETTINGS\_FAIL

### **Description**

The command to set the configuration of the EtherCAT SubDevices encountered an error.

### **Corrective Actions**

Please contact the manufacturer.

## -202: ECAT\_ERR\_MASTER\_GET\_GROUP\_INFO\_FAIL

### **Description**

The command to get the configuration of the EtherCAT group encountered an error.

### **Corrective Actions**

Please contact the manufacturer.

## -203: ECAT\_ERR\_MASTER\_GET\_MASTER\_INFO\_FAIL

### **Description**

The command to get the configuration of the EtherCAT master encountered an error.

### **Corrective Actions**

Please contact the manufacturer.

**-204: ECAT\_ERR\_MASTER\_GET\_DEVICE\_INFO\_FAIL****Description**

The command to get the configuration of the EtherCAT SubDevices encountered an error.

**Corrective Actions**

Please contact the manufacturer.

**-205: ECAT\_ERR\_MASTER\_SET\_GROUP\_SETTINGS\_FAIL****Description**

The command to set the configuration of the EtherCAT group encountered an error.

**Corrective Actions**

Please contact the manufacturer.

**-206: ECAT\_ERR\_MASTER\_MAPPING\_INIT\_FAIL****Description**

Failed to allocate memory for PDO mapping.

**Corrective Actions**

Please contact the manufacturer.

**-207: ECAT\_ERR\_MASTER\_INTERRUPT\_INIT\_FAIL****Description**

Initialization of the interrupt function failed.

**Corrective Actions**

Please contact the manufacturer.

**-208: ECAT\_ERR\_MASTER\_ACTIVE\_FAIL****Description**

The command to activate the EtherCAT master failed.

**Corrective Actions**

Please contact the manufacturer.

## -209: ECAT\_ERR\_MASTER\_ENI\_INITCMDS\_FAIL

### **Description**

The execution of the SDO Download command in the ENI file failed.

### **Corrective Actions**

Please verify the correctness of the ENI file content and ensure that all SubDevices are functioning properly.

## -210: ECAT\_ERR\_MASTER\_NO\_DEVICE

### **Description**

EtherCAT network has no SubDevices.

### **Corrective Actions**

Please connect at least one EtherCAT SubDevice.

## -300: ECAT\_ERR\_MASTER\_ACYCLIC\_INIT\_FAIL

### **Description**

Acyclic transfer interface initialization between dual systems failed.

### **Corrective Actions**

Please contact the manufacturer.

## -301: ECAT\_ERR\_MASTER\_ACYCLIC\_REQUEST\_FAIL

### **Description**

Failed to request acyclic transfer.

### **Corrective Actions**

Please try again later.

**-302: ECAT\_ERR\_MASTER\_ACYCLIC\_BUSY****Description**

Acyclic transfer is busy.

**Corrective Actions**

Please try again later.

**-303: ECAT\_ERR\_MASTER\_ACYCLIC\_TIMEOUT****Description**

Acyclic transfer timed out.

**Corrective Actions**

Please try again later or confirm that the SubDevice is functioning properly.

**-304: ECAT\_ERR\_MASTER\_ACYCLIC\_ERROR****Description**

Acyclic transfer encountered an error.

**Corrective Actions**

Please check the error code. Such as the CoE communication, you can check Abort Code.

**-405: ECAT\_ERR\_MASTER\_ACYCLIC\_WRONG\_STATUS****Description**

Acyclic transfer obtained an invalid status.

**Corrective Actions**

Please contact the manufacturer.

**-400: ECAT\_ERR\_MASTER\_GENERIC\_SEND\_FAIL****Description**

Failed to send data during generic transmission.

**Corrective Actions**

Please contact the manufacturer.

**-401: ECAT\_ERR\_MASTER\_GENERIC\_RECV\_FAIL****Description**

Failed to receive data during generic transmission.

**Corrective Actions**

Please contact the manufacturer.

**-1000: ECAT\_ERR\_MASTER\_NOT\_BEGIN****Description**

The EtherCAT master has not been initialized.

**Corrective Actions**

Please call [EthercatMaster::begin\(\)](#).

**-1001: ECAT\_ERR\_MASTER\_WRONG\_BUFFER\_SIZE****Description**

The size of the input buffer is incorrect.

**Corrective Actions**

Please input a buffer with correct size.

**-1002: ECAT\_ERR\_MASTER\_REDUNDANCY\_NO\_DC****Description**

Cable Redundancy mode does not support DC (Distributed Clocks).

**Corrective Actions**

Do not call this function or avoid Cable Redundancy mode.

**-1003: ECAT\_ERR\_MASTER\_MEMORY\_ALLOCATION\_FAIL****Description**

Failed to allocate memory.

**Corrective Actions**

Please contact the manufacturer.

**-1004: ECAT\_ERR\_MASTER\_OSLAYER\_INIT\_FAIL****Description**

Operating system layer initialization failed.

**Corrective Actions**

Please contact the manufacturer.

**-1005: ECAT\_ERR\_MASTER\_NIC\_INIT\_FAIL****Description**

Network controller initialization failed.

**Corrective Actions**

Please ensure the network controller is present or contact the manufacturer.

**-1006: ECAT\_ERR\_MASTER\_BASE\_INIT\_FAIL****Description**

Initialization of EtherCAT master command interface failed.

**Corrective Actions**

Please contact the manufacturer.

**-1007: ECAT\_ERR\_MASTER\_CIA402\_INIT\_FAIL****Description**

Initialization of EtherCAT master CiA 402 framework failed.

**Corrective Actions**

Please contact the manufacturer.

## -1008: ECAT\_ERR\_MASTER\_SETUP\_PDO\_FAIL

### **Description**

Configuration of PDO mapping for the SubDevice using SDO Download failed.

### **Corrective Actions**

Please ensure that all SubDevices are functioning properly. If the issue is persisted, please contact the manufacturer.

## -1009: ECAT\_ERR\_MASTER\_SCAN\_NETWORK\_FAIL

### **Description**

Failed to scan EtherCAT network.

### **Corrective Actions**

Please connect the network cable properly and ensure that EtherCAT SubDevices are powered on, or verify the presence of the network controller.

## -1010: ECAT\_ERR\_MASTER\_START\_MASTER\_FAIL

### **Description**

Failed to start EtherCAT master.

### **Corrective Actions**

Please ensure that the configuration of PDO mapping for all SubDevices is correct. If the issue is persisted, please contact the manufacturer.

## -1011: ECAT\_ERR\_MASTER\_CYCLETIME\_TOO\_SMALL

### **Description**

The input cycle time is too small.

### **Corrective Actions**

Please try increasing the cycle time.

**-1012: ECAT\_ERR\_MASTER\_DUMP\_OUTPUT\_PDO\_FAIL****Description**

Failed to update output process data using SDO Upload.

**Corrective Actions**

Please ensure that all SubDevices are functioning correctly, and verify that the configuration of PDO mapping is correct.

**-1013: ECAT\_ERR\_MASTER\_CONFIG\_DEVICE\_FAIL****Description**

EtherCAT SubDevice's initialization failed.

**Corrective Actions**

Please contact the manufacturer.

**-1014: ECAT\_ERR\_MASTER\_CONFIG\_MAPPING\_FAIL****Description**

Failed to create PDO mapping.

**Corrective Actions**

Please contact the manufacturer.

**-1015: ECAT\_ERR\_MASTER\_WAIT\_BUS\_SYNC\_TIMEOUT****Description**

Synchronization timeout for all SubDevices.

**Corrective Actions**

Please contact the manufacturer.

**-1016: ECAT\_ERR\_MASTER\_WAIT\_MASTER\_SYNC\_TIMEOUT****Description**

Synchronization timeout between MDevice and SubDevices.

**Corrective Actions**

Please contact the manufacturer.

**-1017: ECAT\_ERR\_MASTER\_CYCLIC\_START\_FAIL****Description**

Failed to start cyclic transmission.

**Corrective Actions**

Please contact the manufacturer.

**-1018: ECAT\_ERR\_MASTER\_WRONG\_BUFFER\_POINTER****Description**

Incorrect data buffer pointer.

**Corrective Actions**

Please input the correct data buffer pointer.

**-1050: ECAT\_ERR\_MASTER\_ENI\_INIT\_FAIL****Description**

ENI initialization failed.

**Corrective Actions**

Please confirm that the specified ENI file exists. If this issue persists, please contact the manufacturer.

## -1051: ECAT\_ERR\_MASTER\_ENI\_MISMATCH

### **Description**

The SubDevice information in the ENI file does not match the scanned EtherCAT network SubDevices.

### **Corrective Actions**

Please adjust the order of the SubDevices on the EtherCAT network or configure the SubDevice identifications.

## -1100: ECAT\_ERR\_MASTER\_STOPPED

### **Description**

EtherCAT MDevice has not been started.

### **Corrective Actions**

Please call [EthercatMaster::start\(\)](#) or avoid calling this function.

## -1101: ECAT\_ERR\_MASTER\_STARTED

### **Description**

EtherCAT MDevice has already been started.

### **Corrective Actions**

Please call [EthercatMaster::stop\(\)](#) or avoid calling this function.

## -1102: ECAT\_ERR\_MASTER\_NOT\_IN\_PREOP

### **Description**

EtherCAT MDevice is not in PRE-OP state.

### **Corrective Actions**

Please ensure that all SubDevices are in PRE-OP state before calling this function.

## -1103: ECAT\_ERR\_MASTER\_NOT\_IN\_SAFEOP

### **Description**

EtherCAT MDevice is not in SAFE-OP state.

### **Corrective Actions**

Please ensure that all SubDevices are in SAFE-OP state before calling this function.

## -1104: ECAT\_ERR\_MASTER\_NOT\_IN\_OP

### **Description**

EtherCAT MDevice is not in OP state.

### **Corrective Actions**

Please ensure that all SubDevices are in OP state before calling this function.

## -1200: ECAT\_ERR\_MASTER\_II\_TRANSITION\_FAIL

### **Description**

Switching all SubDevices to INIT state during EtherCAT MDevice initialization failed.

### **Corrective Actions**

Please power cycle all SubDevices and then run the EtherCAT MDevice program again.

## -1201: ECAT\_ERR\_MASTER\_IP\_TRANSITION\_FAIL

### **Description**

Failed to transition EtherCAT state from INIT to PRE-OP.

### **Corrective Actions**

Please ensure that all SubDevices are functioning properly, or check the quality of the network connections.

## -1202: ECAT\_ERR\_MASTER\_PS\_TRANSITION\_FAIL

### **Description**

Failed to transition EtherCAT state from PRE-OP to SAFE-OP.

### **Corrective Actions**

Please check the correctness of the PDO mapping configuration for all SubDevices. If DC synchronization is enabled, try adjusting the related parameters for DC synchronization.

## -1203: ECAT\_ERR\_MASTER\_SO\_TRANSITION\_FAIL

### **Description**

Failed to transition EtherCAT state from SAFE-OP to OP.

### **Corrective Actions**

Please try adjusting the related parameters for DC synchronization and test again.

## -2000: ECAT\_ERR\_DEVICE\_NOT\_EXIST

### **Description**

The specified SubDevice does not exist.

### **Corrective Actions**

Please do not access the specified SubDevice.

## -2001: ECAT\_ERR\_DEVICE\_NOT\_ATTACH

### **Description**

The object of such SubDevice has not been initialized.

### **Corrective Actions**

Please call [attach\(\)](#) to initialize the SubDevice object.

## -2002: ECAT\_ERR\_DEVICE\_NO\_MAILBOX

### **Description**

The SubDevice does not support Mailbox.

### **Corrective Actions**

Please do not call Mailbox-related functions for the SubDevice.

## -2003: ECAT\_ERR\_DEVICE\_NO\_DC

### **Description**

The SubDevice does not support DC synchronization.

### **Corrective Actions**

Please do not call DC-related functions for the SubDevice.

## -2004: ECAT\_ERR\_DEVICE\_WRONG\_INPUT

### **Description**

Incorrect input parameter.

### **Corrective Actions**

Please input the correct parameter.

## -2005: ECAT\_ERR\_DEVICE\_MEMORY\_ALLOCATION\_FAIL

### **Description**

Failed to allocate memory for the SubDevice object.

### **Corrective Actions**

Please contact the manufacturer.

## -2006: ECAT\_ERR\_DEVICE\_VENDOR\_ID\_MISMATCH

### **Description**

The Vendor ID of the SubDevice does not match that of the SubDevice object.

### **Corrective Actions**

Please use the correct SubDevice object.

## -2007: ECAT\_ERR\_DEVICE\_PRODUCT\_CODE\_MISMATCH

### **Description**

The Product Code of the SubDevice does not match that of the SubDevice object.

### **Corrective Actions**

Please use the correct SubDevice object.

## -2008: ECAT\_ERR\_DEVICE\_NO\_SUCH\_FUNCTION

### **Description**

The SubDevice does not support this feature.

### **Corrective Actions**

Please do not call this function.

## -2009: ECAT\_ERR\_DEVICE\_FUNCTION\_NOT\_INIT

### **Description**

The specific function of the SubDevice has not been initialized.

### **Corrective Actions**

Please initialize that function.

## -2010: ECAT\_ERR\_DEVICE\_BUSY

### **Description**

The SubDevice is busy.

### **Corrective Actions**

Please try again later.

## -2011: ECAT\_ERR\_DEVICE\_TIMEOUT

### **Description**

The SubDevice has encountered a timeout.

### **Corrective Actions**

Please try again later or confirm that the SubDevice is functioning properly.

## -2012: ECAT\_ERR\_DEVICE\_NO\_DATA

### **Description**

The SubDevice has no available data.

### **Corrective Actions**

Please try again later.

## -2100: ECAT\_ERR\_DEVICE\_SII\_READ\_FAIL

### **Description**

Failed to read the SII EEPROM of the SubDevice.

### **Corrective Actions**

Please ensure that this function is called when the EtherCAT state of the SubDevice is INIT or PRE-OP.

## -2101: ECAT\_ERR\_DEVICE\_SII\_WRITE\_FAIL

### **Description**

Failed to write the SII EEPROM of the SubDevice.

### **Corrective Actions**

Please ensure that this function is called when the EtherCAT state of the SubDevice is INIT or PRE-OP.

## -2200: ECAT\_ERR\_DEVICE\_PDO\_NOT\_EXIST

### **Description**

The SubDevice has no output process data.

### **Corrective Actions**

Please do not call this function.

## -2201: ECAT\_ERR\_DEVICE\_PDO\_OUT\_OF\_RANGE

### **Description**

Accessing beyond the process data range of the SubDevice.

### **Corrective Actions**

Please access the correct range of process data for the SubDevice.

## -2300: ECAT\_ERR\_DEVICE\_FOE\_NOT\_SUPPORT

### **Description**

The SubDevice does not support FoE.

### **Corrective Actions**

Please do not call FoE-related functions for the SubDevice.

**-2310: ECAT\_ERR\_DEVICE\_FOE\_REQUEST\_FAIL****Description**

Failed to request FoE communication.

**Corrective Actions**

Please try again later.

**-2311: ECAT\_ERR\_DEVICE\_FOE\_TIMEOUT****Description**

FoE communication timeout.

**Corrective Actions**

Please verify that the SubDevice supports FoE and is functioning properly.

**-2312: ECAT\_ERR\_DEVICE\_FOE\_ERROR****Description**

FoE communication encountered an error.

**Corrective Actions**

Please verify that the SubDevice supports FoE and is functioning properly.

**-2313: ECAT\_ERR\_DEVICE\_FOE\_BUFFER\_TOO\_SMALL****Description**

The size of the input buffer for FoE communication is too small.

**Corrective Actions**

Please input a buffer with suitable size.

**-2314: ECAT\_ERR\_DEVICE\_FOE\_READ\_FAIL****Description**

Failed to read the file via FoE communication.

**Corrective Actions**

Please verify that the SubDevice supports reading files via FoE communication.

**-2315: ECAT\_ERR\_DEVICE\_FOE\_WRITE\_FAIL****Description**

Failed to write the file via FoE communication.

**Corrective Actions**

Please verify that the SubDevice supports writing files via FoE communication.

**-2400: ECAT\_ERR\_DEVICE\_COE\_SDO\_NOT\_SUPPORT****Description**

The SubDevice does not support SDO commands of CoE communication.

**Corrective Actions**

Please do not call functions related to SDO commands of CoE communication.

**-2401: ECAT\_ERR\_DEVICE\_COE\_SDO\_INFO\_NOT\_SUPPORT****Description**

The SubDevice does not support SDO Information commands of CoE communication.

**Corrective Actions**

Please do not call functions related to SDO Information commands of CoE communication.

**-2410: ECAT\_ERR\_DEVICE\_COE\_BUSY****Description**

CoE communication is busy.

**Corrective Actions**

Please try again later.

**-2411: ECAT\_ERR\_DEVICE\_COE\_REQUEST\_FAIL****Description**

Failed to request CoE communication.

**Corrective Actions**

Please try again later.

**-2412: ECAT\_ERR\_DEVICE\_COE\_TIMEOUT****Description**

CoE communication timeout.

**Corrective Actions**

Please try again later or confirm that the SubDevice is functioning properly.

**-2413: ECAT\_ERR\_DEVICE\_COE\_ERROR****Description**

CoE communication encountered an error.

**Corrective Actions**

Please check the abort code for CoE communication.

**-2500: ECAT\_ERR\_DEVICE\_CIA402\_NOT\_EXIST****Description**

This SubDevice does not have objects related to CiA 402.

**Corrective Actions**

Please do not call functions related to CiA 402 for the SubDevice.

**-2501: ECAT\_ERR\_DEVICE\_CIA402\_ADD\_FAIL****Description**

Failed to insert the CiA 402 SubDevice object to the CiA 402 framework of EtherCAT MDevice.

**Corrective Actions**

Please confirm whether the CiA 402 SubDevice object has been successfully inserted into the CiA 402 framework of the EtherCAT MDevice.

## -2502: ECAT\_ERR\_DEVICE\_CIA402\_TYPE\_MISMATCH

### **Description**

The content of the object of Device Type (Index 1000h) for the SubDevice is not CiA 402 type.

### **Corrective Actions**

Please do not call functions related to CiA 402 for the SubDevice.

## -2503: ECAT\_ERR\_DEVICE\_CIA402\_NO\_MODE\_SUPPORT

### **Description**

This CiA 402 SubDevice does not support any operation modes defined by CiA 402.

### **Corrective Actions**

Please do not call functions related to CiA 402 for the SubDevice. Also, ensure that the CiA 402 SubDevice supports CiA 402.

## -2504: ECAT\_ERR\_DEVICE\_CIA402\_WRONG\_MODE

### **Description**

This function cannot be called in the current CiA 402 operation mode for the SubDevice.

### **Corrective Actions**

Please change the CiA 402 operation mode of this SubDevice to the correct mode before calling this function.

## -2505: ECAT\_ERR\_DEVICE\_CIA402\_MODE\_NOT\_SUPPORT

### **Description**

The SubDevice does not support the specified CiA 402 operation mode.

### **Corrective Actions**

Please do not change the CiA 402 operation mode of this SubDevice to an unsupported mode.

**-2506: ECAT\_ERR\_DEVICE\_CIA402\_CHANGE\_WRONG\_STATE****Description**

The current CiA 402 state does not allow switching to the specified CiA 402 state.

**Corrective Actions**

Please check the current CiA 402 state. If it is in FAULT state, switch to SWITCH\_ON\_DISABLED state first, and then switch to the target state.

**-2507: ECAT\_ERR\_DEVICE\_CIA402\_WRITE\_OBJECT\_FAIL****Description**

Accessing CiA 402 objects is not allowed using SDO Upload or SDO Download in the cyclic callback functions.

**Corrective Actions**

Please avoid using SDO Upload/Download to access CiA 402 objects in the cyclic callback functions. If needed, map the CiA 402 objects to process data.

**-2580: ECAT\_ERR\_DEVICE\_CIA402\_NO\_SUCH\_TOUCH\_PROBE****Description**

The input number for the Touch Probe functionality is incorrect.

**Corrective Actions**

Please input the correct Touch Probe number, ranging from 0 to 1.

**-2581: ECAT\_ERR\_DEVICE\_CIA402\_NO\_SUCH\_TOUCH\_PROBE\_SOURCE****Description**

The input signal source for the Touch Probe functionality is incorrect.

**Corrective Actions**

Please input the correct signal source, ranging from 0 to 2.

**-2600: ECAT\_ERR\_DEVICE\_EOE\_NOT\_SUPPORT****Description**

The SubDevice does not support EoE.

**Corrective Actions**

Please do not call EoE-related functions for the SubDevice.

**-2601: ECAT\_ERR\_DEVICE\_EOE\_NO\_SUCH\_PORT****Description**

Incorrect EoE port number.

**Corrective Actions**

Please input the correct EoE port number.

**-2602: ECAT\_ERR\_DEVICE\_EOE\_TOO\_MUCH\_CONTENT****Description**

The input content is too much.

**Corrective Actions**

Please provide the correct content.

**-2610: ECAT\_ERR\_DEVICE\_EOE\_BUSY****Description**

EoE communication is busy.

**Corrective Actions**

Please try again later.

**-2611: ECAT\_ERR\_DEVICE\_EOE\_REQUEST\_FAIL****Description**

Failed to request EoE communication.

**Corrective Actions**

Please try again later.

**-2612: ECAT\_ERR\_DEVICE\_EOE\_TIMEOUT****Description**

EoE communication timeout.

**Corrective Actions**

Please verify that the SubDevice supports EoE and is functioning properly.

**-3000: ECAT\_ERR\_GROUP\_WRONG\_INPUT****Description**

The input parameter is incorrect.

**Corrective Actions**

Please input the correct parameter.

**-3001: ECAT\_ERR\_GROUP\_NOT\_ATTACH****Description**

The object of such group has not been initialized.

**Corrective Actions**

Please initialize the group object.

## A.3 Error Callback Code

Error Callback Code list:

ECAT_ERR_WKC_SINGLE_FAULT	Working Counter Fault.	2000001
ECAT_ERR_WKC_MULTIPLE_FAULTS	Working Counter Multiple Faults.	2000002
ECAT_ERR_SINGLE_LOST_FRAME	Single Lost Frame.	2000003
ECAT_ERR_MULTIPLE_LOST_FRAMES	Multiple Lost Frames.	2000004
ECAT_ERR_LOST_SLAVE	Lost SubDevice.	2000005
ECAT_ERR_STATE_MISMATCH	State Mismatch.	2000006
ECAT_ERR_CABLE_BROKEN	Cable Broken.	2000007
ECAT_ERR_WAIT_ACK_TIMEOUT	Wait ACK Timeout.	2001000

## A.4 Event Callback Code

Event Callback Code list:

ECAT_EVT_STATE_CHANGED	State Changed.	1000001
ECAT_EVT_CABLE_RECONNECTED	Cable Reconnected.	1000002

## A.5 SDO Abort Code

The CoE SDO Abort Codes defined in ETG.1000.6:

Value	Meaning
0x05030000	Toggle bit not changed.
0x05040000	SDO protocol timeout.
0x05040001	Client/Server command specifier not valid or unknown.
0x05040005	Out of memory.
0x06010000	Unsupported access to an object.
0x06010001	Attempt to read to a write only object.
0x06010002	Attempt to write to a read only object.
0x06010003	Subindex cannot be written, SIO must be 0 for write access.
0x06010004	SDO Complete access not supported for objects of variable length such as ENUM object types.
0x06010005	Object length exceeds mailbox size.
0x06010006	Object mapped to RxPDO, SDO Download blocked.
0x06020000	The object does not exist in the object directory.
0x06040041	The object cannot be mapped into the PDO.
0x06040042	The number and length of the objects to be mapped would exceed the PDO length.
0x06040043	General parameter incompatibility reason.
0x06040047	General internal incompatibility in the device.
0x06060000	Access failed due to a hardware error.
0x06070010	Data type does not match; length of service parameter does not match.
0x06070012	Data type does not match, length of service parameter too high.
0x06070013	Data type does not match, length of service parameter too low.
0x06090011	Subindex does not exist.
0x06090030	Value range of parameter exceeded (only for write access).
0x06090031	Value of parameter written too high.
0x06090032	Value of parameter written too low.
0x06090036	Maximum value is less than minimum value.
0x08000000	General error.
0x08000020	Data cannot be transferred or stored to the application. NOTE: This is the general Abort Code in case no further detail on the reason can be determined. It is recommended to use one of the more detailed Abort Codes. (0x08000021, 0x08000022)
0x08000021	Data cannot be transferred or stored to the application because of local control. NOTE: "local control" means an application specific reason. It does not mean the ESM-specific control.

0x08000022	Data cannot be transferred or stored to the application because of the present device state. NOTE: "device state" means the ESM state.
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present.

The extended CoE SDO Abort Codes defined in ETG.1020:

Value	Meaning
0x06010003	Subindex cannot be written, SIO must be 0 for write access.
0x06010004	SDO Complete access not supported for objects of variable length such as ENUM object types.
0x06010005	Object length exceeds mailbox size.
0x06010006	Object mapped to RxPDO, SDO Download blocked. This optional Abort Code is used only in states SafeOp and Op.
0x06090033	configured module list does not match detected module list. It shall be used if Object 0xF03x is written but does not fit to object 0xF05x.

## A.6 Data Type

The Basic Data Types defined in ETG.1000.6:

Index (hex)	Object Type	Name
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	INTEGER8
0003	DEFTYPE	INTEGER16
0004	DEFTYPE	INTEGER32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000A	DEFTYPE	OCTET_STRING
000B	DEFTYPE	UNICODE_STRING
000C	DEFTYPE	TIME_OF_DAY
000D	DEFTYPE	TIME_DIFFERENCE
000F	DEFTYPE	DOMAIN
0010	DEFTYPE	INTEGER24
0011	DEFTYPE	REAL64
0012	DEFTYPE	INTEGER40
0013	DEFTYPE	INTEGER48
0014	DEFTYPE	INTEGER56
0015	DEFTYPE	INTEGER64
0016	DEFTYPE	UNSIGNED24
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001A	DEFTYPE	UNSIGNED56
001B	DEFTYPE	UNSIGNED64
001D	DEFTYPE	GUID
001E	DEFTYPE	BYTE
002D	DEFTYPE	BITARR8
002E	DEFTYPE	BITARR16
002F	DEFTYPE	BITARR32

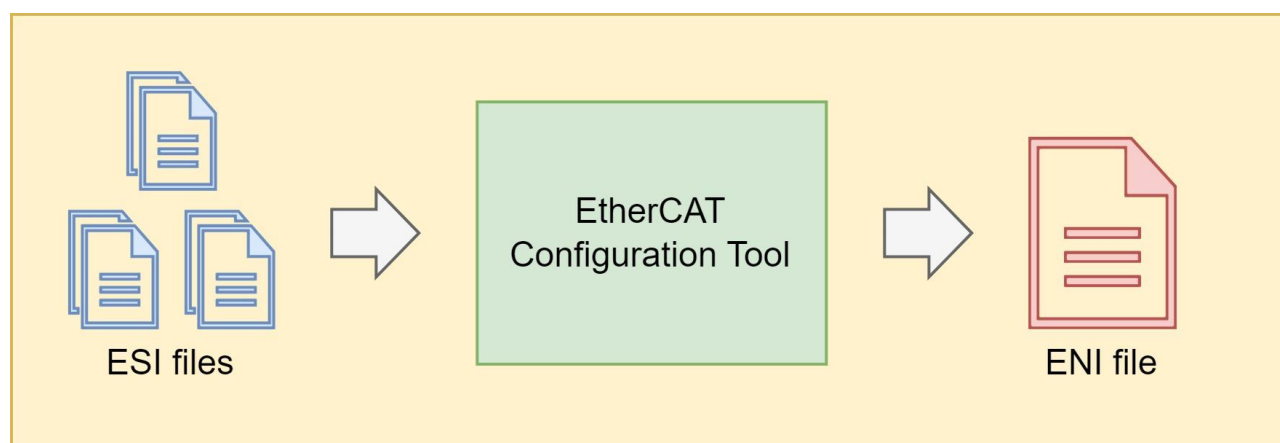
The Extended Data Types defined in ETG.1000.6:

<b>Index (hex)</b>	<b>Object Type</b>	<b>Name</b>
0021	DEFSTRUCT	PDO_MAPPING
0023	DEFSTRUCT	IDENTITY
0025	DEFSTRUCT	COMMAND_PAR
0029	DEFSTRUCT	SYNC_PAR
0030	DEFTYPE	BIT1
0031	DEFTYPE	BIT2
0032	DEFTYPE	BIT3
0033	DEFTYPE	BIT4
0034	DEFTYPE	BIT5
0035	DEFTYPE	BIT6
0036	DEFTYPE	BIT7
0037	DEFTYPE	BIT8
0040-005F	DEFSTRUCT	Manufacturer Specific Complex Data Types
0060-007F	DEFTYPE	Device Profile 0 Specific Standard Data Types
0080-009F	DEFSTRUCT	Device Profile 0 Specific Complex Data Types
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types

## A.7 EtherCAT Network Information

The EtherCAT Network Information (ENI) contains the necessary settings to configure an EtherCAT network. The XML-based file contains general information about the MDevice and the configurations of every SubDevice connected to the MDevice.

The EtherCAT Configuration Tool reads the ESI files or online scans the network for all SubDevices, then user can configure relevant EtherCAT settings, such as PDO mapping and enabling DC, and then export the ENI file.



The EtherCAT Technology Group specifies that the EtherCAT MDevice Software must support at least one of the following in the Network Configuration section: *Online Scanning* or *Reading ENI*. This library, however, supports both. In the case of *Reading ENI*, this library currently extracts only partial information from the ENI file for network configuration.

The extracted information includes the following:

#### EtherCATConfig : Config : SubDevice : Info

- **Elements**

- VendorId
- ProductCode

- **Attribute**

- Identification : Value

- **Purpose**

Used to check whether the EtherCAT SubDevices on the network match the SubDevices specified in the ENI file. The checking rules are as follows:

- Check if the number of SubDevices in the ENI file matches the number of SubDevices on the network.
- For SubDevices in the ENI file with the Identification: Value attribute, check if there are SubDevices on the network with matching Alias Address and Identification: Value attribute, as well as Vendor ID and Product Code. If such SubDevices exist, it indicates a successful match.
- For SubDevices with the Identification: Value attribute that fail to match, or those without this attribute, check if the Vendor ID and Product Code of the SubDevice with the same sequence number on the network match.

#### EtherCATConfig : Config : SubDevice : Mailbox

- **Elements**

- Send : MailboxSendInfoType : Start
- Recv : MailboxRecvInfoType : Start

- **Purpose**

Used to configure the mailbox Physical Start Address of an EtherCAT device.

#### EtherCATConfig : Config : SubDevice : Mailbox : CoE

- **Elements**

- InitCmds : InitCmd : Index
- InitCmds : InitCmd : SubIndex
- InitCmds : InitCmd : Data
- InitCmds : InitCmd : Timeout

- **Attribute**

- InitCmds : InitCmd : CompleteAccess

- **Purpose**

After switching the EtherCAT state machine to the Pre-Operational state, execute the CoE initialization commands for the EtherCAT SubDevice in [EthercatMaster::begin\(\)](#).

**EtherCATConfig : Config : SubDevice : ProcessData****• Elements**

- Recv : BitLength
- Send : BitLength

**• Purpose**

The bit length of the output process data and input process data of an EtherCAT SubDevice is provided to the firmware for relevant configuration.

**EtherCATConfig : Config : SubDevice : ProcessData : Sm****• Elements**

- SyncManagerSettings : StartAddress
- SyncManagerSettings : ControlByte
- SyncManagerSettings : Enable

**• Purpose**

Used to configure the Sync Manager registers for the process data of an EtherCAT device.

**EtherCATConfig : Config : SubDevice: DC****• Elements**

- CycleTime0
- CycleTime1
- ShiftTime

**• Purpose**

Used to configure the DC parameters of an EtherCAT device.

# Warranty

This product is warranted to be in good working order for a period of one year from the date of purchase. Should this product fail to be in good working order at any time during this period, we will, at our option, replace or repair it at no additional charge except as set forth in the following terms. This warranty does not apply to products damaged by misuse, modifications, accident or disaster. Vendor assumes no liability for any damages, lost profits, lost savings or any other incidental or consequential damage resulting from the use, misuse of, originality to use this product. Vendor will not be liable for any claim made by any other related party. Return authorization must be obtained from the vendor before returned merchandise will be accepted. Authorization can be obtained by calling or faxing the vendor and requesting a Return Merchandise Authorization (RMA) number. Returned goods should always be accompanied by a clear problem description.

All Trademarks appearing in this manuscript are registered trademark of their respective owners. All Specifications are subject to change without notice.

©ICOP Technology Inc. 2026