User Manual

EtherCAT CiA402 API

86Duino Coding IDE 501+

EtherCAT Library - EthercatDevice_CiA402 Class

(Version1.1)

REVISION

DATE	VERSION	DESCRIPTION
2024/12/20	**Version1.0**	**New Release.**
2025/1/3	**Version1.1**	**Replace 'Master' by 'MainDevice' or 'MDevice', and 'Slave' by 'SubDevice', and will show the terms MainDevice and SubordinateDevice in the list of abbreviations.**

COPYRIGHT

Ver.1.1 January, 2025

TRADEMARKS ACKNOWLEDGMENT

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

- Global: www.icop.com.tw
- USA: www.icoptech.com
- Japan: www.icop.co.jp
- Europe: www.icoptech.eu
- China: www.icop.com.cn

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

- 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 About QEC MainDevice (MDevice)

QEC MDevice is an EtherCAT MDevice compatible with 86Duino Coding IDE 501+. It offers real-time EtherCAT communication between EtherCAT MDevice and EtherCAT SubDevice. 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 MDevice. 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 SubDevice scanning
- Import ENI file
- Setting EtherCAT MDevice
- Configure EtherCAT SubDevice

For other detailed functions, please refer to the 86EVA User Manual.

1.1.2 QEC EtherCAT MDevice 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

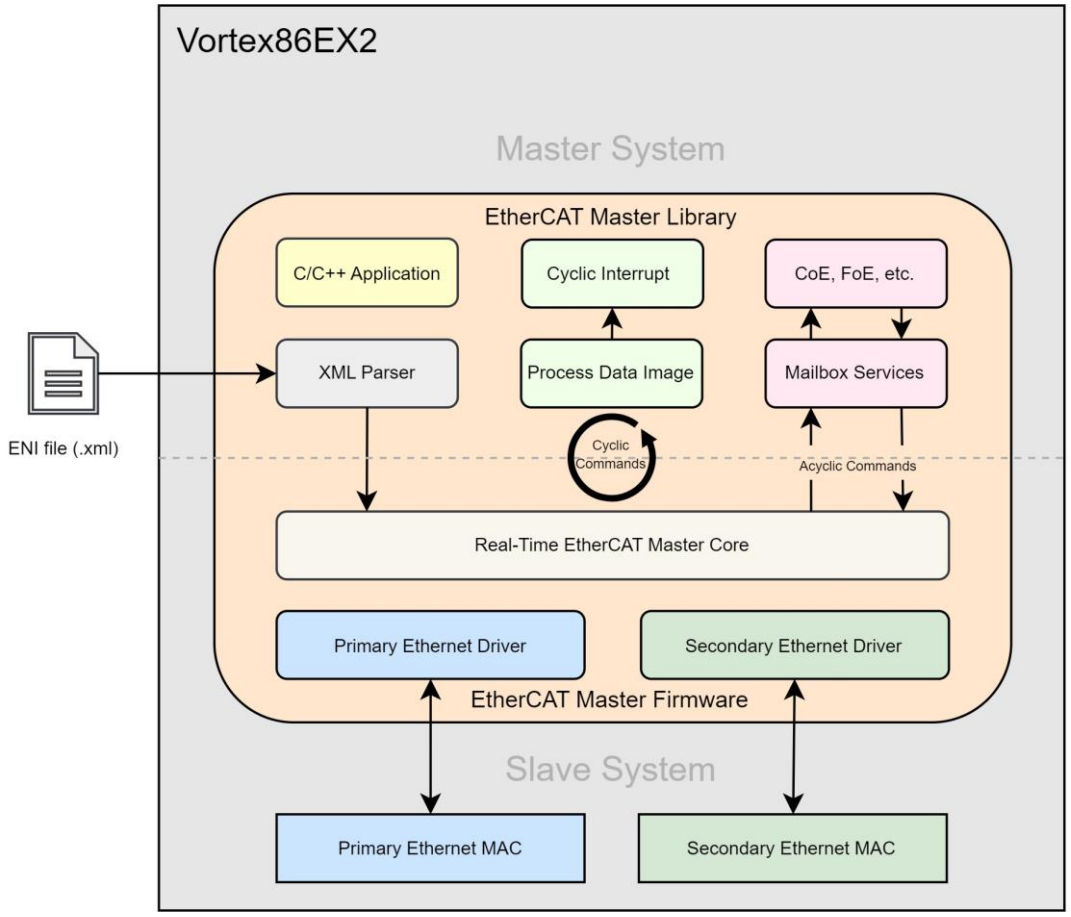
The programs are designed to run on the FreeDOS operating system and have been compiled using the GCC compiler provided by the DJGPP environment.

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

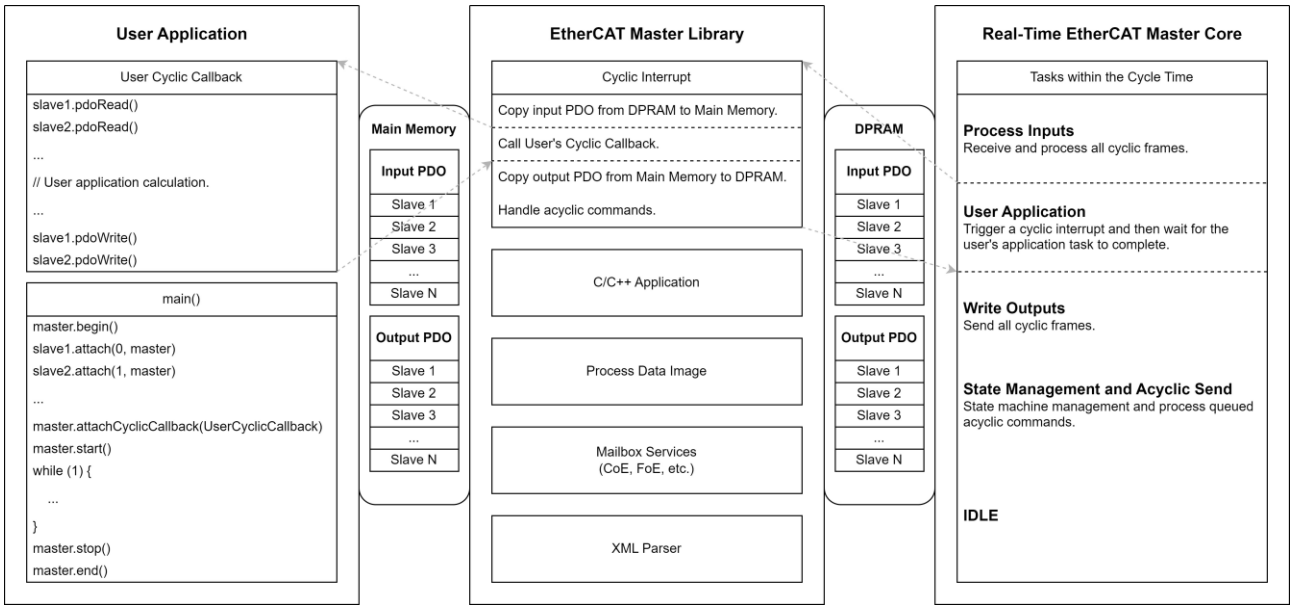
The primary focus of this section is the synchronization of dual-system PDO data. As illustrated in the diagram below, the User Application and EtherCAT MDevice Library blocks run on the Master System, while the Real-Time EtherCAT MDevice Core runs on the Slave System.

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.

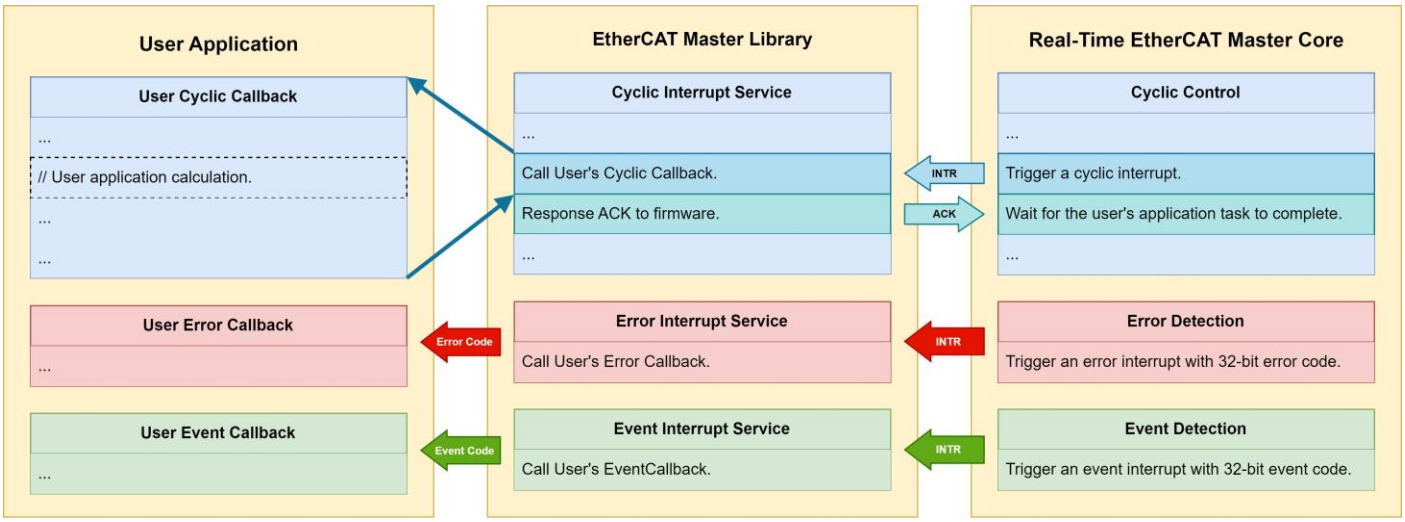
Upon reaching the **User Application** stage, the EtherCAT MDevice Core triggers a Cyclic Interrupt to the Master System. Upon receiving the Cyclic Interrupt, the Master System executes the interrupt handling procedure of the EtherCAT MDevice Library. It moves Input PDO data from DPRAM to Main Memory, 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 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.



1.1.5 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.

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

1.2.1 Feature Table

Word Usage:

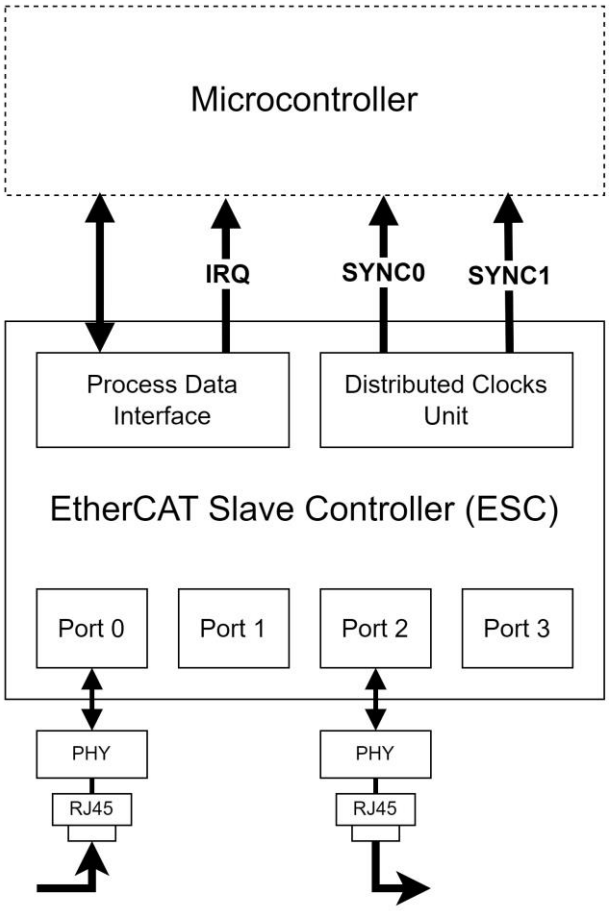
- **O** equals supported.

Feature Name	Short Description	QEC MDevice
Basic Features		
Service Commands	Support of all commands	O
SubDevice with Device Emulation	Support SubDevice with and without application controller	O
EtherCAT State Machine	Support of ESM special behavior	O
Error Handling	Checking of network or SubDevice errors, e.g. Working Counter	O
EtherCAT Frame Types	Support EtherCAT Frames	O
Process Data Exchange		
Cyclic PDO	Cyclic process data exchange	O
Network Configuration		
Online scanning	Network configuration functionality included in EtherCAT MDevice	O
Reading ENI	Network Configuration taken from ENI file	O
Compare Network configuration	Compare configured and existing network configuration during boot-up	O
Explicit Device Identification	Identification used for Hot Connect and prevention against cable swapping	O
Access to EEPROM	Support routines to access EEPROM via ESC register	O
Mailbox Support		
Support Mailbox	Main functionality for mailbox transfer	O
Mailbox Resilient Layer	Support underlying resilient layer	O
CAN application layer over EtherCAT (CoE)		
SDO Up/Download	Normal and expedited transfer	O

Segmented Transfer	Segmented transfer	O
Complete Access	Transfer the entire object (with all sub-indices) at once	O
SDO Info service	Services to read object dictionary	O
FoE		
FoE Protocol	Support FoE Protocol	O
Firmware Up/Download	Password, FileName should be given by the application	O
Synchronization with Distributed Clock (DC)		
DC support	Support of Distributed Clock	O

1.3 Synchronization

The time synchronization among all SubDevice 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 SubDevice with DC. For a more detailed explanation of DC, please refer to Distributed Clocks.



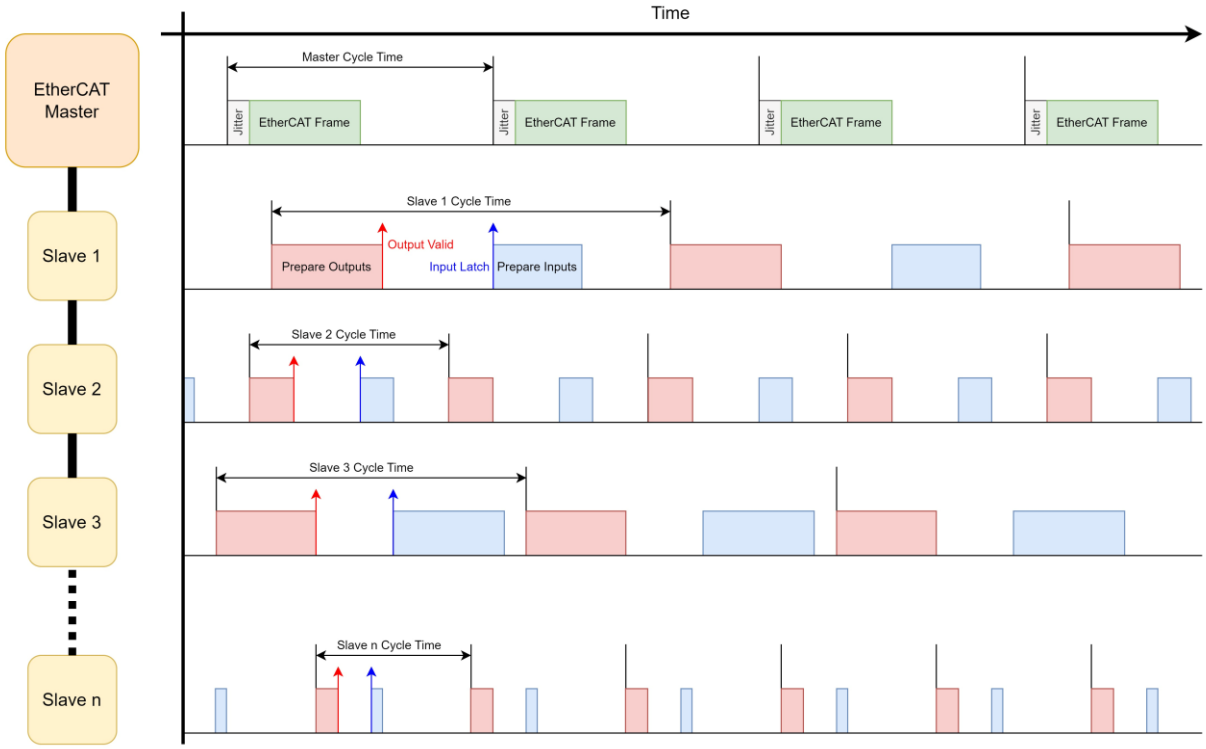
The ESC has three synchronization output pins: ***IRQ***, ***SYNC0***, and ***SYNC1***. The IRQ pin generates a signal to the upper-layer microcontroller (µC) after the ESC receives EtherCAT Cyclic Frames. SYNC0 and SYNC1 pins cyclically generate signals to the µC based on the configuration in the DC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

There are three synchronization modes in EtherCAT:

- Free Run
- SM-Synchronous
- DC-Synchronous

1.3.1 Free Run

The EtherCAT MDevice and all EtherCAT SubDevice 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.

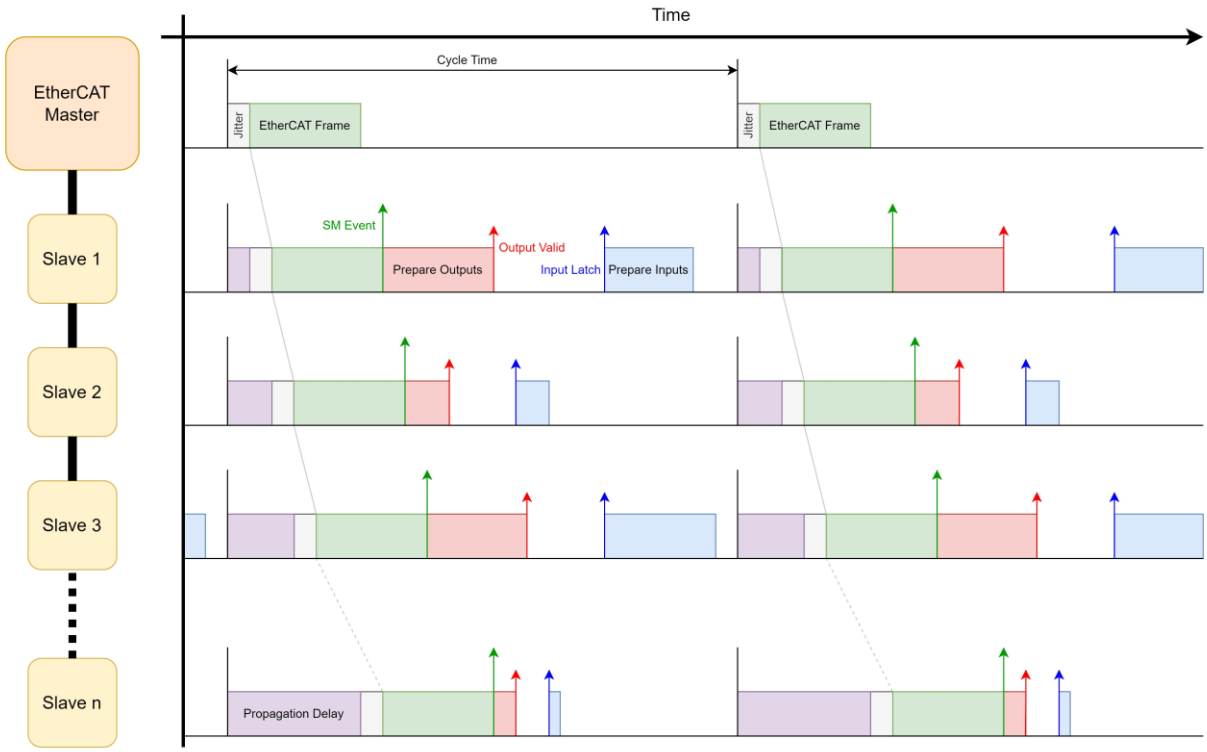


1.3.2 SM-Synchronous

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 µC. As shown in the diagram below, cyclic frames are received by SubDevice with the same jitter of MDevice in sending them. Even assuming zero jitter, due to finite hardware ***Propagation Delay*** the last SubDevice 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 SubDevice, 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 SubDevice to be close to each other. However, due to the inability to calculate the propagation delays, the adjustment is quite challenging.



1.3.3 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 µC. 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.

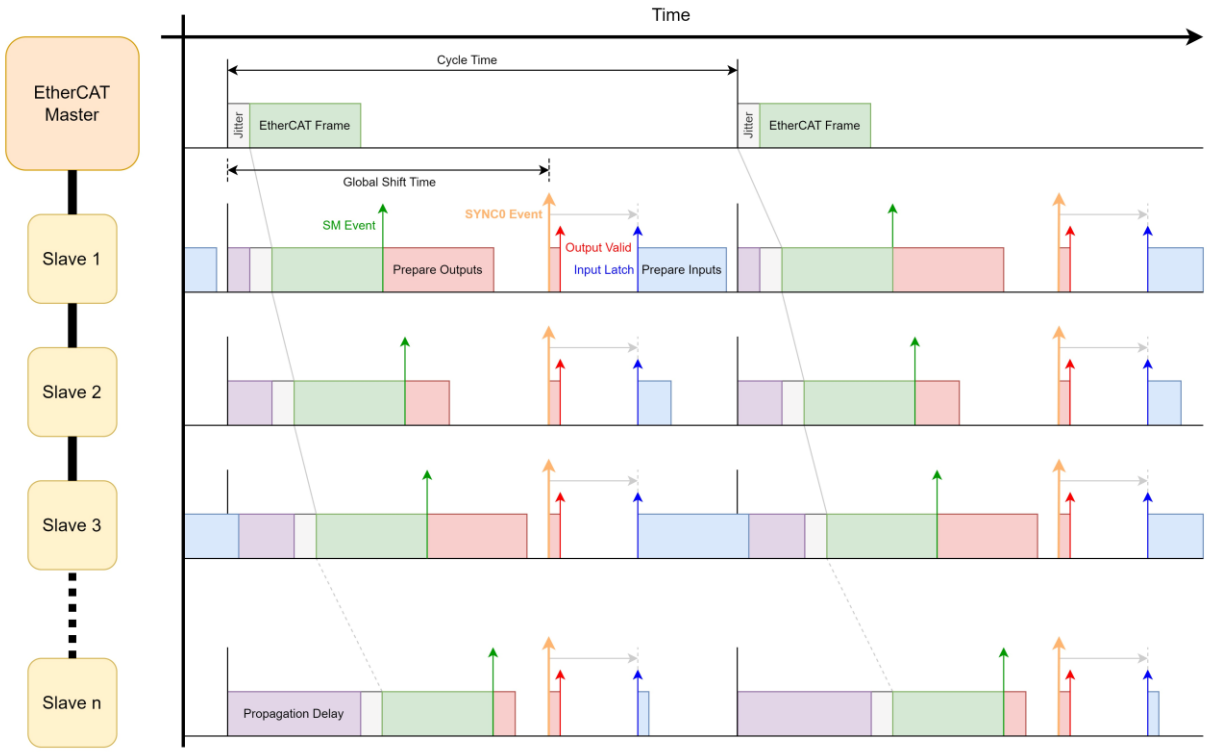
Because SYNC0 events are triggered by the DC unit and all SYNC0 events among the SubDevice have almost no offset, thanks to the periodic sending of APRW/FPRW commands to synchronize the system time of all SubDevice, the accuracy can reach the ***nanosecond*** level.

If the SubDevice 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 SubDevice 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 SubDevice.
- 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 µC to execute *Prepare Outputs* when the SM event occurs
    - Trigger µC 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*.



1.4 About 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:

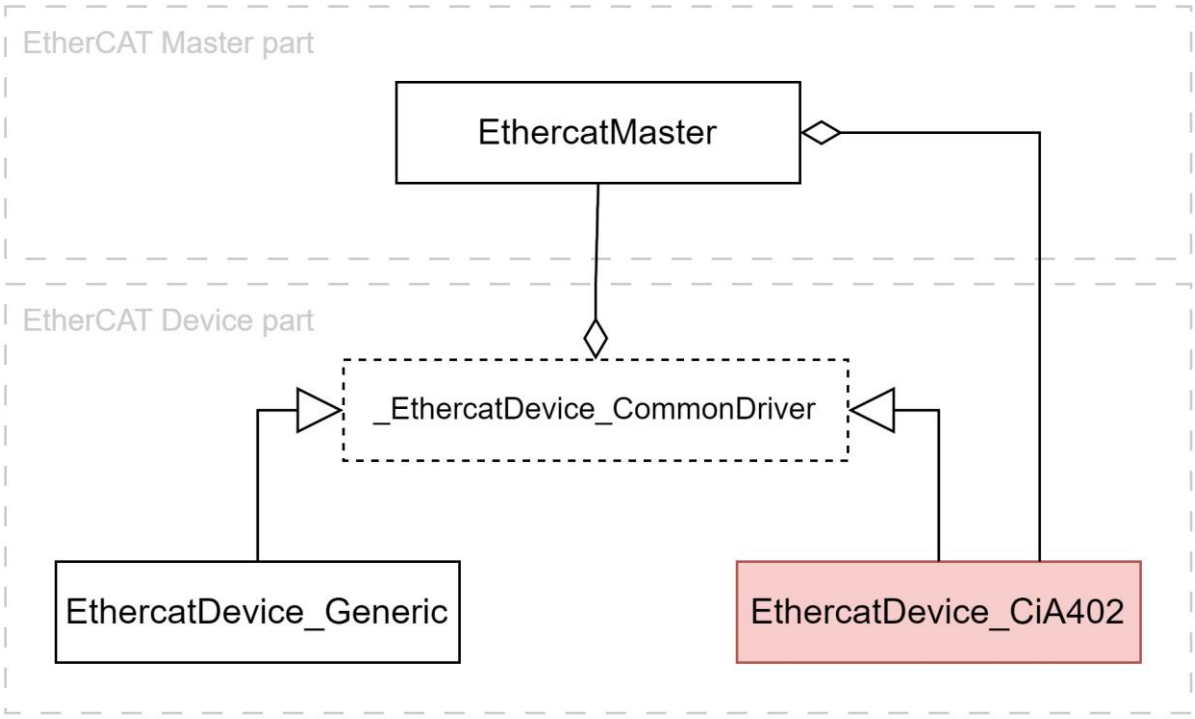
- **Operation Modes**
  - Profile Position (pp)
  - Profile Velocity (pv)
  - Profile Torque (tq)
  - Homing (hm)
  - Cyclic Synchronous Position (csp)
  - Cyclic Synchronous Velocity (csv)
  - Cyclic Synchronous Torque (cst)
- **Function Groups**
  - Touch Probe

Implementation Directive for CiA402 Drive Profile.
Directive for using IEC 61800-7-201 within EtherCAT-based servo drives.

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.

For more detailed information about EtherCAT Device Class, please refer to EtherCAT Library API User Manual - QEC.

1.4.1 Functions Table

This section will briefly introduce the EthercatDevice_CiA402 API usage list.

Function Name	Description	Callback Available
Initialization-related functions		
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.	O
Control-related functions		
getCiA402Mode()	Get the current mode of operation. (6061h)	O[1]
setCiA402Mode()	Switch the mode of operation. (6060h, 6061h, 6502h)	O[1, 2]
getCiA402State()	Get the current CiA 402 state. (6041h)	O
setCiA402State()	Switch the CiA 402 state. (6040h, 6041h)	O[2]
enable()	Enable the drive function and power on the motor. (6040h, 6041h)	O[2]
disable()	Disable the drive function and power off the motor. (6040h, 6041h)	O[2]
Operation-related functions		
setTargetPosition()	Set the target position. (607Ah)	O[1]
setTargetVelocity()	Set the target velocity. (60FFh)	O[1]
setTargetTorque()	Set the target torque. (6071h)	O[1]
setProfileAcceleration()	Set the profile acceleration. (6083h)	O[1]
setProfileDeceleration()	Set the profile deceleration. (6084h)	O[1]
setMaxAcceleration()	Set the max acceleration. (60C5h)	O[1]
setMaxDeceleration()	Set the max deceleration. (60C6h)	O[1]
setMaxProfileVelocity()	Set the max profile velocity. (607Fh)	O[1]
setMotionProfileType()	Set the motion profile type. (6086h)	O[1]
setPositionWindow()	Set the position window. (6067h)	O[1]
setPositionWindowTime()	Set the position window time. (6068h)	O[1]
setPositionOffset()	Set the position offset. (60B0h)	O[1]
setSoftwarePositionLimit()	Set the software position limit. (607Dh)	O[1]
setFollowingErrorWindow()	Set the following error window. (6065h)	O[1]
setPositionPolarity()	Set the position polarity. (607Eh)	O[1]
setVelocityWindow()	Set the velocity window. (606Dh)	O[1]
setVelocityWindowTime()	Set the velocity window time. (606Eh)	O[1]
setVelocityThreshold()	Set the velocity threshold. (606Fh)	O[1]
setVelocityOffset()	Set the velocity offset. (60B1h)	O[1]
setMaxMotorSpeed()	Set the max motor speed. (6080h)	O[1]
setVelocityPolarity()	Set the velocity polarity. (607Eh)	O[1]
setTorqueOffset()	Set the torque offset. (60B2h)	O[1]

setMaxTorque()	Set the max torque. (6072h)	O[1]
setPositiveTorqueLimit()	Set the positive torque limit. (60E0h)	O[1]
setNegativeTorqueLimit()	Set the negative torque limit. (60E1h)	O[1]
setQuickStopDeceleration()	Set the quick stop deceleration. (6085h)	O[1]
setQuickStopOptionCode()	Set the quick stop option code. (605Ah)	
setShutdownOptionCode()	Set the shutdown option code. (605Bh)	
setDisableOperationOptionCode()	Set the disable operation option code. (605Ch)	
setHaltOptionCode()	Set the halt option code. (605Dh)	
setFaultReactionOptionCode()	Set the fault reaction option code. (605Eh)	
getErrorCode()	Get the error code. (603Fh)	O[1]
getSupportedDriveModes()	Get the supported drive modes. (6502h)	O[1]
getMotorResolution()	Get the motor resolution. (60EFh)	
getPositionActualValue()	Get the position actual value. (6064h)	O[1]
getVelocityActualValue()	Get the velocity actual value. (606Ch)	O[1]
getTorqueActualValue()	Get the torque actual value. (6077h)	O[1]
getCurrentActualValue()	Get the current actual value. (6078h)	O[1]
getPositionDemandValue()	Get the position demand value. (6062h)	O[1]
getPositionDemandInternalValue()	Get the position demand internal value. (60FCh)	O[1]
getPositionActualInternalValue()	Get the position actual internal value. (6063h)	O[1]
getAdditionalPositionActualValue()	Get the additional position actual value. (60E4h)	O[1]
getFollowingErrorActualValue()	Get the following error actual value. (60F4h)	O[1]
getVelocityDemandValue()	Get the velocity demand value. (606Bh)	O[1]
getTorqueDemandValue()	Get the torque demand value. (6074h)	O[1]
getDigitalInputs()	Get the digital inputs. (60FDh)	O[1]
Profile Position mode (pp) related functions		
pp_SetVelocity()	Set the profile velocity. (6081h)	
pp_SetAcceleration()	Set the profile acceleration. (6083h)	
pp_SetDeceleration()	Set the profile deceleration. (6084h)	
pp_SetMotionProfileType()	Set the motion profile type. (6086h)	
pp_Run()	Move to the target position. (6040h, 6041h, 607Ah)	
pp_IsTargetReached()	Check if the target position has been reached. (6041h)	O
pp_CheckFollowingError()	Check if the following error occurs. (6041h)	O
pp_Halt()	Pause the current operation. (6040h, 6041h)	
pp_Resume()	Resume the paused operation. (6040h, 6041h)	
Profile Velocity mode (pv) related functions		
pv_SetAcceleration()	Set the profile acceleration. (6083h)	
pv_SetDeceleration()	Set the profile deceleration. (6084h)	
pv_SetMotionProfileType()	Set the motion profile type. (6086h)	
pv_Run()	Move at a target velocity continuously. (6041h, 60FFh)	
pv_IsTargetReached()	Check if the target velocity has been reached. (6041h)	O
pv_CheckZeroSpeed()	Check if the speed is zero. (6041h)	O

pv_CheckMaxSlippageError()	Check if the maximum slippage error occurs. (6041h)	O
pv_Halt()	Pause the current operation. (6040h, 6041h)	
pv_Resume()	Resume the paused operation. (6040h, 6041h)	
Profile Torque mode (tq) related functions		
tq_SetTorqueSlope()	Set the torque slope. (6087h)	
tq_SetTorqueProfileType()	Set the torque profile type. (6088h)	
tq_SetMotorRatedCurrent()	Set the motor rated current. (6075h)	
tq_SetMotorRatedTorque()	Set the motor rated torque. (6076h)	
tq_Run()	Drive continuously at the target torque. (6041h, 6071h)	
tq_IsTargetReached()	Check if the target torque has been reached. (6041h)	O
tq_Halt()	Pause the current operation. (6040h, 6041h)	
tq_Resume()	Resume the paused operation. (6040h, 6041h)	
Homing mode (hm) related functions		
hm_SetHomeOffset()	Set the home offset. (607Ch)	
hm_SetHomingMethod()	Set the homing method. (6098h)	
hm_SetHomingSpeeds()	Set the homing speeds. (6099h)	
hm_SetHomingAcceleration()	Set the homing acceleration. (609Ah)	
hm_Run()	Initiate a homing operation. (6040h, 6041h)	
hm_IsAttained()	Check the status of the homing operation. (6041h)	O
hm_Stop()	Stop the homing operation. (6040h, 6041h)	
Function Group "Touch Probe" related functions		
enableTouchProbe1()	Enable the touch probe 1. (60B8h, 60B9h, 60D0h)	
enableTouchProbe2()	Enable the touch probe 2. (60B8h, 60B9h, 60D0h)	
disableTouchProbe1()	Disable the touch probe 1. (60B8h, 60B9h)	
disableTouchProbe2()	Disable the touch probe 2. (60B8h, 60B9h)	
isTouchProbe1ValueReady()	Check if a positive or negative edge has occurred on the touch probe 1 signal. (60B9h)	O[1]
isTouchProbe2ValueReady()	Check if a positive or negative edge has occurred on the touch probe 2 signal. (60B9h)	O[1]
readTouchProbe1Value()	Read the touch probe position 1. (60BAh, 60BBh)	O[1]
readTouchProbe2Value()	Read the touch probe position 2. (60BCh, 60BDh)	O[1]
Low-level functions for mode-specific flow control		
setHaltBit()	Set the halt bit in the controlword. (6040h)	O
isTargetReached()	Check if the target has reached. (6041h)	O
setModeSpecificBit4()	Set the bit 4 in the controlword. (6040h)	O
setModeSpecificBit5()	Set the bit 5 in the controlword. (6040h)	O
setModeSpecificBit6()	Set the bit 6 in the controlword. (6040h)	O
checkModeSpecificBit12()	Check the value of bit 12 in the statusword. (6041h)	O
checkModeSpecificBit13()	Check the value of bit 13 in the statusword. (6041h)	O

- **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.

Ch. 2

Functions

86Duino Coding IDE 501+.

EtherCAT Library - EthercatDevice_CiA402 Class

2.1 Initialization-related Functions

Initialization-related functions for the EthercatDevice_CiA402 class.

Functions:

- attach()
- detach()
- isStepper()

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);
int attach(uint16_t slave_id, uint8_t drive_id, EthercatMaster *master, EthercatAttachMode mode = ECAT_SLAVE_NO);
int attach(uint16_t slave_id, uint8_t drive_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] drive_id`
  The ID of CiA 402 drive in the EtherCAT SubDevice. Typically used when a single EtherCAT SubDevice has multiple CiA 402 drive axes.
- `[in] master`
  The object of EthercatMaster class to which it should be attached.
- `[in] mode`
  The definition of slave_id :
  I. `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.
  II. `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.

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()`.

- WARNING: Prohibited from being called in the callback functions.

Example Code

- Single-axis CiA 402 EtherCAT Servo Drive

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

}
```

- Multi-axis CiA 402 EtherCAT Servo Drive

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor1;
EthercatDevice_CiA402 motor2;
EthercatDevice_CiA402 motor3;

void setup() {
  master.begin();
  motor1.attach(0, 0, master);
  motor2.attach(0, 1, master);
  motor3.attach(0, 2, master);
}

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

}
```

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 `attach()`.

- WARNING: Prohibited from being called in the callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  delay(1000);
  motor.detach();
}

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

}
```

isStepper()

Description

Check if the EtherCAT SubDevice is a stepper motor drive.

Syntax

```
int isStepper();
```

Parameters

None.

Return Value

Return whether the EtherCAT SubDevice is a stepper motor drive. 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_CiA402 motor;

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

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

  Serial.print("Stepper Motor Drive: ");
  Serial.println(motor.isStepper());
}

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

}
```

2.2 Control-related Functions

Control-related functions for the EthercatDevice_CiA402 class.

Functions:

- getCiA402Mode()
- setCiA402Mode()
- getCiA402State()
- setCiA402State()
- enable()
- disable()

getCiA402Mode()

Description

Get the current mode of operation.

Relevant Objects

- Object 6061_h: Modes of operation display

Syntax

```
int getCiA402Mode();
```

Parameters

None.

Return Value

Return the current mode of operation:

Definition	Code	Description
CIA402_PP_MODE	1	Profile Position mode (pp)
CIA402_PV_MODE	3	Profile Velocity mode (pv)
CIA402_TQ_MODE	4	Profile Torque mode (tq)
CIA402_HOMING_MODE	6	Homing mode (hm)
CIA402_CSP_MODE	8	Cyclic Synchronous Position mode (csp)
CIA402_CSV_MODE	9	Cyclic Synchronous Velocity mode (csv)
CIA402_CST_MODE	10	Cyclic Synchronous Torque mode (cst)

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  Serial.print("Operation Mode: ");
  Serial.println(motor.getCiA402Mode());
}
```

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

}
```

setCiA402Mode()

Description

Switch the mode of operation.

Relevant Objects

- Object 6060_h: Modes of operation
- Object 6061_h: Modes of operation display
- Object 6502_h: Supported drive modes

Syntax

```
int setCiA402Mode(int mode, uint32_t timeout_ms = 1000);
```

Parameters

- `[in] mode`
  The mode of operation to be switched:

Definition	Code	Description
CIA402_PP_MODE	1	Profile Position mode (pp)
CIA402_PV_MODE	3	Profile Velocity mode (pv)
CIA402_TQ_MODE	4	Profile Torque mode (tq)
CIA402_HOMING_MODE	6	Homing mode (hm)
CIA402_CSP_MODE	8	Cyclic Synchronous Position mode (csp)
CIA402_CSV_MODE	9	Cyclic Synchronous Velocity mode (csv)
CIA402_CST_MODE	10	Cyclic Synchronous Torque mode (cst)

- `[in] timeout_ms`
  Timeout in milliseconds.

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 used in callback functions if the related object is mapped to a PDO. However, when used in a callback, this function will ignore the timeout parameter and will not wait for the actual value to match the set value.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
```

```
}

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

}
```

getCiA402State()

Description

Get the current CiA 402 state.

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int getCiA402State();
```

Parameters

None.

Return Value

Return the current CiA 402 state:

Definition	Code	State
CIA402_NOT_READY_TO_SWITCH_ON	0	Not ready to switch on.
CIA402_SWITCH_ON_DISABLED	1	Switch on disabled.
CIA402_READY_TO_SWITCH_ON	2	Ready to switch on.
CIA402_SWITCHED_ON	3	Switched on.
CIA402_OPERATION_ENABLED	4	Operation enabled.
CIA402_FAULT	5	Fault.
CIA402_FAULT_REACTION_ACTIVE	6	Fault reaction active.
CIA402_QUICK_STOP_ACTIVE	7	Quick stop active.

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

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
```

```
  Serial.print("CiA402 State: ");
  Serial.println(motor.getCiA402State());
  delay(1000);
}
```

setCiA402State()

Description

Switch the CiA 402 state.

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int setCiA402State(int state, uint32_t timeout_ms = 1000);
```

Parameters

- `[in] mode`
  The CiA 402 state to be switched:

Definition	Code	State
CIA402_SWITCH_ON_DISABLED	1	Switch on disabled.
CIA402_READY_TO_SWITCH_ON	2	Ready to switch on.
CIA402_SWITCHED_ON	3	Switched on.
CIA402_OPERATION_ENABLED	4	Operation enabled.
CIA402_QUICK_STOP_ACTIVE	7	Quick stop active.

- `[in] timeout_ms`
  Timeout in milliseconds.

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 will wait for the actual value to match the set value until a timeout occurs. When used in a callback function, it becomes non-blocking and will ignore the timeout parameter, not waiting for the actual value to match the set value.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  motor.setCiA402State(CIA402_OPERATION_ENABLED);
  delay(4000);
```

```
  motor.setCiA402State(CIA402_SWITCH_ON_DISABLED);
}

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

}
```

enable()

Description

Enable the drive function and power on the motor. The behavior of this function is identical to setCiA402State(CIA402_OPERATION_ENABLED).

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int enable(uint32_t timeout_ms = 1000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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 will wait for the actual value to match the set value until a timeout occurs. When used in a callback function, it becomes non-blocking and will ignore the timeout parameter, not waiting for the actual value to match the set value.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  motor.enable();
}

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

}
```

disable()

Description

Disable the drive function and power off the motor. The behavior of this function is identical to `setCiA402State(CIA402_SWITCH_ON_DISABLED)`.

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int disable(uint32_t timeout_ms = 1000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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 will wait for the actual value to match the set value until a timeout occurs. When used in a callback function, it becomes non-blocking and will ignore the timeout parameter, not waiting for the actual value to match the set value.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  motor.enable();
  delay(4000);
  motor.disable();
}

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

2.3 Operation-related Functions

Operation-related functions for the EthercatDevice_CiA402 class.

Functions:

- setTargetPosition()
- setTargetVelocity()
- setTargetTorque()
- setProfileAcceleration()
- setProfileDeceleration()
- setMaxAcceleration()
- setMaxDeceleration()
- setMaxProfileVelocity()
- setMotionProfileType()
- setPositionWindow()
- setPositionWindowTime()
- setPositionOffset()
- setSoftwarePositionLimit()
- setFollowingErrorWindow()
- setPositionPolarity()
- setVelocityWindow()
- setVelocityWindowTime()
- setVelocityThreshold()
- setVelocityOffset()
- setMaxMotorSpeed()
- setVelocityPolarity()
- setTorqueOffset()
- setMaxTorque()
- setPositiveTorqueLimit()
- setNegativeTorqueLimit()
- setQuickStopDeceleration()
- setQuickStopOptionCode()
- setShutdownOptionCode()
- setDisableOperationOptionCode()
- setHaltOptionCode()
- setFaultReactionOptionCode()
- getErrorCode()
- getSupportedDriveModes()
- getMotorResolution()
- getPositionActualValue()

- getVelocityActualValue()
- getTorqueActualValue()
- getCurrentActualValue()
- getPositionDemandValue()
- getPositionDemandInternalValue()
- getPositionActualInternalValue()
- getAdditionalPositionActualValue()
- getFollowingErrorActualValue()
- getVelocityDemandValue()
- getTorqueDemandValue()
- getDigitalInputs()

setTargetPosition()

Description

Set the target position object.

Relevant Objects

- Object $607A_h$: Target position

Syntax

```
int setTargetPosition(int32_t value);
```

Parameters

- `[in] value`
  The target position to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
  motor.setTargetPosition(10000);
}

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

}
```

setTargetVelocity()

Description

Set the target velocity object.

Relevant Objects

- Object $60FF_h$: Target velocity

Syntax

```
int setTargetVelocity(int32_t value);
```

Parameters

- `[in] value`
  The target velocity to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
  motor.setTargetVelocity(1000);
}

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

}
```

setTargetTorque()

Description

Set the target torque object.

Relevant Objects

- Object 6071_h: Target torque

Syntax

```
int setTargetTorque(int16_t value);
```

Parameters

- `[in] value`
  The target torque to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
  motor.setTargetTorque(30);
}

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

}
```

setProfileAcceleration()

Description

Set the profile acceleration object.

Relevant Objects

- Object 6083_h: Profile acceleration

Syntax

```
int setProfileAcceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile acceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setProfileAcceleration(5000);
}

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

}
```

setProfileDeceleration()

Description

Set the profile deceleration object.

Relevant Objects

- Object 6084_h: Profile deceleration

Syntax

```
int setProfileDeceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile deceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setProfileDeceleration(5000);
}

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

}
```

setMaxAcceleration()

Description

Set the max acceleration object.

Relevant Objects

- Object $60C5_h$: Max acceleration

Syntax

```
int setMaxAcceleration(uint32_t value);
```

Parameters

- `[in] value`
  The max acceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMaxAcceleration(200);
}

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

}
```

setMaxDeceleration()

Description

Set the max deceleration object.

Relevant Objects

- Object $60C6_h$: Max deceleration

Syntax

```
int setMaxDeceleration(uint32_t value);
```

Parameters

- `[in] value`
  The max deceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMaxDeceleration(200);
}

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

}
```

setMaxProfileVelocity()

Description

Set the max profile velocity object.

Relevant Objects

- Object 607F$_h$: Max profile velocity

Syntax

```
int setMaxProfileVelocity(uint32_t value);
```

Parameters

- `[in] value`
  The max profile velocity to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMaxProfileVelocity(50000);
}

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

}
```

setMotionProfileType()

Description

Set the motion profile type object.

Relevant Objects

- Object 6086_h: Motion profile type

Syntax

```
int setMotionProfileType(int16_t value);
```

Parameters

- `[in] value`

  The motion profile type to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Linear ramp (trapezoidal profile)
1	sin^2 ramp
2	Jerk-free ramp
3	Jerk-limited ramp
4 .. 32767	Reserved

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMotionProfileType(0);
}

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

setPositionWindow()

Description

Set the position window object.

Relevant Objects

- Object 6067_h: Position window

Syntax

```
int setPositionWindow(uint32_t value);
```

Parameters

- `[in] value`
  The position window to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setPositionWindow(100);
}

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

}
```

setPositionWindowTime()

Description

Set the position window time object.

Relevant Objects

- Object 6068_h : Position window time

Syntax

```
int setPositionWindowTime(uint16_t value);
```

Parameters

- `[in] value`
  The position window time to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setPositionWindowTime(10);
}

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

}
```

setPositionOffset()

Set the position offset object.

Relevant Objects

- Object $60B0_h$: Position offset

Syntax

```
int setPositionOffset(int32_t value);
```

Parameters

- `[in] value`
  The position offset to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setPositionOffset(10000);
}

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

}
```

setSoftwarePositionLimit()

Description

Set the software position limit object.

Relevant Objects

- Object $607D_h$: Software position limit

Syntax

```
int setSoftwarePositionLimit(int32_t min_value, int32_t max_value);
```

Parameters

- `[in] min_value`
  The minimum software position limit to be set.
- `[in] max_value`
  The maximum software position limit to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setSoftwarePositionLimit(-2147483648, 2147483647);
}

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

}
```

setFollowingErrorWindow()

Description

Set the following error window object.

Relevant Objects

- Object 6065_h: Following error window

Syntax

```
int setFollowingErrorWindow(uint32_t value);
```

Parameters

- `[in] value`
  The following error window to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setFollowingErrorWindow(3840000);
}

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

}
```

setPositionPolarity()

Description

Set the position polarity object.

Relevant Objects

- Object $607E_h$: Polarity

Syntax

```
int setPositionPolarity(int polarity);
```

Parameters

- `[in] polarity`

  The position polarity to be set:

Definition	Code	Description
CIA402_POLARITY_POSITIVE	0	multiply by 1
CIA402_POLARITY_NEGATIVE	1	multiply by -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 non-blocking and can be called in callback functions if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setPositionPolarity(CIA402_POLARITY_POSITIVE);
}

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

}
```

setVelocityWindow()

Description

Set the velocity window object.

Relevant Objects

- Object $606D_h$: Velocity window

Syntax

```
int setVelocityWindow(uint16_t value);
```

Parameters

- `[in] value`
  The velocity window to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setVelocityWindow(100);
}

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

}
```

setVelocityWindowTime()

Description

Set the velocity window time object.

Relevant Objects

- Object 606E$_h$: Velocity window time

Syntax

```
int setVelocityWindowTime(uint16_t value);
```

Parameters

- `[in] value`
  The velocity window time to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setVelocityWindowTime(10);
}

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

}
```

setVelocityThreshold()

Description

Set the velocity threshold object.

Relevant Objects

- Object 606F$_h$: Velocity threshold

Syntax

```
int setVelocityThreshold(uint16_t value);
```

Parameters

- `[in] value`
  The velocity threshold to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setVelocityThreshold(100);
}

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

}
```

setVelocityOffset()

Description

Set the velocity offset object.

Relevant Objects

- Object $60B1_h$: Velocity offset

Syntax

```
int setVelocityOffset(int32_t value);
```

Parameters

- `[in] value`
  The velocity offset to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setVelocityOffset(1000);
}

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

}
```

setMaxMotorSpeed()

Description

Set the max motor speed object.

Relevant Objects

- Object 6080_h: Max motor speed

Syntax

```
int setMaxMotorSpeed(uint32_t value);
```

Parameters

- `[in] value`
  The max motor speed to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMaxMotorSpeed(5000);
}

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

}
```

setVelocityPolarity()

Description

Set the velocity polarity object.

Relevant Objects

- Object $607E_h$: Polarity

Syntax

```
int setVelocityPolarity(int polarity);
```

Parameters

- [in] polarity

  The velocity polarity to be set:

Definition	Code	Description
CIA402_POLARITY_POSITIVE	0	multiply by 1
CIA402_POLARITY_NEGATIVE	1	multiply by -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 non-blocking and can be called in callback functions if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setVelocityPolarity(CIA402_POLARITY_POSITIVE);
}

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

}
```

setTorqueOffset()

Description

Set the torque offset object.

Relevant Objects

- Object $60B2_h$: Torque offset

Syntax

```
int setTorqueOffset(int16_t value);
```

Parameters

- `[in] value`
  The torque offset to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setTorqueOffset(500);
}

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

}
```

setMaxTorque()

Description

Set the max torque object.

Relevant Objects

- Object 6072_h: Max torque

Syntax

```
int setMaxTorque(uint16_t value);
```

Parameters

- `[in] value`
  The max torque to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setMaxTorque(3000);
}

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

}
```

setPositiveTorqueLimit()

Description

Set the positive torque limit object.

Relevant Objects

- Object $60E0_h$: Positive torque limit value

Syntax

```
int setPositiveTorqueLimit(uint16_t value);
```

Parameters

- `[in] value`
  The positive torque limit to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setPositiveTorqueLimit(3000);
}

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

}
```

setNegativeTorqueLimit()

Description

Set the negative torque limit object.

Relevant Objects

- Object $60E1_h$: Negative torque limit value

Syntax

```
int setNegativeTorqueLimit(uint16_t value);
```

Parameters

- `[in] value`
  The negative torque limit to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setNegativeTorqueLimit(3000);
}

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

}
```

setQuickStopDeceleration()

Description

Set the quick stop deceleration object.

Relevant Objects

- Object 6085_h: Quick stop deceleration

Syntax

```
int setQuickStopDeceleration(uint32_t value);
```

Parameters

- `[in] value`
  The quick stop deceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setQuickStopDeceleration(200);
}

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

}
```

setQuickStopOptionCode()

Description

Set the quick stop option code object.

Relevant Objects

- Object $605A_h$: Quick stop option code

Syntax

```
int setQuickStopOptionCode(int16_t value);
```

Parameters

- `[in] value`

  The quick stop option code to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Disable drive function
1	Slow down on slow down ramp
2	Slow down on quick stop ramp
3	Slow down on the current limit
4	Slow down on the voltage limit
5	Slow down on slow down ramp and stay in QUICK STOP
6	Slow down on quick stop ramp and stay in QUICK STOP
7	Slow down on the current limit and stay in QUICK STOP
8	Slow down on the voltage limit and stay in QUICK STOP
9 ... 32767	Reserved

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_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setQuickStopOptionCode(1);
}
```

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

}
```

setShutdownOptionCode()

Description

Set the shutdown option code object.

Relevant Objects

- Object $605B_h$: Shutdown option code

Syntax

```
int setShutdownOptionCode(int16_t value);
```

Parameters

- `[in] value`

  The shutdown option code to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Disable drive function
1	Slow down with slow down ramp; disable of the drive function
2 ... 32767	Reserved

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_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setShutdownOptionCode(1);
}

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

}
```

setDisableOperationOptionCode()

Description

Set the disable operation option code object.

Relevant Objects

- Object $605C_h$: Disable operation option code

Syntax

```
int setDisableOperationOptionCode(int16_t value);
```

Parameters

- [in] value

  The disable operation option code to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Disable drive function
1	Slow down with slow down ramp and then disabling of the drive function
2 ... 32767	Reserved

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_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setDisableOperationOptionCode(1);
}

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

}
```

setHaltOptionCode()

Description

Set the halt option code object.

Relevant Objects

- Object $605D_h$: Halt option code

Syntax

```
int setHaltOptionCode(int16_t value);
```

Parameters

- `[in] value`

  The halt option code to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Disable drive, motor is free to rotaten
1	Slow down on slow down ramp
2	Slow down on quick stop ramp
3	Slow down on the current limit
4	Slow down on the voltage limit
5 ... 32767	Reserved

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_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setHaltOptionCode(1);
}

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

setFaultReactionOptionCode()

Description

Set the fault reaction option code object.

Relevant Objects

- Object $605E_h$: Fault reaction option code

Syntax

```
int setFaultReactionOptionCode(int16_t value);
```

Parameters

- `[in] value`

  The fault reaction option code to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Disable drive, motor is free to rotaten
1	Slow down on slow down ramp
2	Slow down on quick stop ramp
3	Slow down on the current limit
4	Slow down on the voltage limit
5 ... 32767	Reserved

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_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setFaultReactionOptionCode(1);
}

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

getErrorCode()

Description

Get the error code object.

Relevant Objects

- Object $603F_h$: Error code

Syntax

```
uint16_t getErrorCode();
```

Parameters

None.

Return Value

Return the value of the error code object.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Error Code: ");
  Serial.println(motor.getErrorCode());
  delay(1000);
  // ...
}
```

getSupportedDriveModes()

Description

Get the supported drive modes object.

Relevant Objects

- Object 6502_h: Supported drive modes

Syntax

```
uint32_t getSupportedDriveModes();
```

Parameters

None.

Return Value

Return the value of the supported drive modes object:

Bit	Description
0	Profile Position mode (pp)
1	Velocity mode (vl)
2	Profile Velocity mode (pv)
3	Profile Torque mode (tq)
4	Reserved
5	Homing mode (hm)
6	Interpolated Position mode (ip)
7	Cyclic synchronous position mode (csp)
8	Cyclic synchronous velocity mode (csv)
9	Cyclic synchronous torque mode (cst)
10	Cyclic synchronous torque mode with commutation angle (cstca)
11 ... 15	Reserved
16 ... 31	Manufacturer Specific

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
```

```
  motor.attach(0, master);
  Serial.print("Supported Drive Modes: ");
  Serial.println(motor.getSupportedDriveModes());
}

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

}
```

getMotorResolution()

Description

Get the motor resolution object.

Relevant Objects

- Object $60EF_h$: Motor resolution

Syntax

```
uint32_t getMotorResolution();
```

Parameters

None.

Return Value

Return the value of the motor resolution object.

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_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  Serial.print("Motor Resolution: ");
  Serial.println(motor.getMotorResolution());
}

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

}
```

getPositionActualValue()

Description

Get the position actual value object.

Relevant Objects

- Object 6064_h: Position actual value

Syntax

```
int32_t getPositionActualValue();
```

Parameters

None.

Return Value

Return the position actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Position: ");
  Serial.println(motor.getPositionActualValue());
  delay(1000);
  // ...
}
```

getVelocityActualValue()

Description

Get the velocity actual value object.

Relevant Objects

- Object $606C_h$: Velocity actual value

Syntax

```
int32_t getVelocityActualValue();
```

Parameters

None.

Return Value

Return the velocity actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Velocity: ");
  Serial.println(motor.getVelocityActualValue());
  delay(1000);
  // ...
}
```

getTorqueActualValue()

Description

Get the torque actual value object.

Relevant Objects

- Object 6077_h: Torque actual value

Syntax

```
int16_t getTorqueActualValue();
```

Parameters

None.

Return Value

Return the torque actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Torque: ");
  Serial.println(motor.getTorqueActualValue());
  delay(1000);
  // ...
}
```

getCurrentActualValue()

Description

Get the current actual value object.

Relevant Objects

- Object 6078_h: Current actual value

Syntax

```
int16_t getCurrentActualValue();
```

Parameters

None.

Return Value

Return the current actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Current: ");
  Serial.println(motor.getCurrentActualValue());
  delay(1000);
  // ...
}
```

getPositionDemandValue()

Description

Get the position demand value object.

Relevant Objects

- Object 6062_h: Position demand value

Syntax

```
int32_t getPositionDemandValue();
```

Parameters

None.

Return Value

Return the position demand value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Position: ");
  Serial.println(motor.getPositionDemandValue());
  delay(1000);
  // ...
}
```

getPositionDemandInternalValue()

Description

Get the position demand internal value object.

Relevant Objects

- Object $60FC_h$: Position demand internal value

Syntax

```
int32_t getPositionDemandInternalValue();
```

Parameters

None.

Return Value

Return the position demand internal value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Position: ");
  Serial.println(motor.getPositionDemandValue());
  delay(1000);
  // ...
}
```

getPositionActualInternalValue()

Description

Get the position actual internal value object.

Relevant Objects

- Object 6063_h: Position actual internal value

Syntax

```
int32_t getPositionActualInternalValue();
```

Parameters

None.

Return Value

Return the position actual internal value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Position: ");
  Serial.println(motor.getPositionActualInternalValue());
  delay(1000);
  // ...
}
```

getAdditionalPositionActualValue()

Description

Get the additional position actual value object.

Relevant Objects

- Object $60E4_h$: Additional position actual value

Syntax

```
int32_t getAdditionalPositionActualValue(int ordinal = 1);
```

Parameters

None.

Return Value

Return the additional position actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Position: ");
  Serial.println(motor.getAdditionalPositionActualValue());
  delay(1000);
  // ...
}
```

getFollowingErrorActualValue()

Description

Get the following error actual value object.

Relevant Objects

- Object 60F4$_h$h: Following error actual value

Syntax

```
int32_t getFollowingErrorActualValue();
```

Parameters

None.

Return Value

Return the following error actual value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Following Error: ");
  Serial.println(motor.getFollowingErrorActualValue());
  delay(1000);
  // ...
}
```

getVelocityDemandValue()

Description

Get the velocity demand value object.

Relevant Objects

- Object $606B_h$: Velocity demand value

Syntax

```
int32_t getVelocityDemandValue();
```

Parameters

None.

Return Value

Return the velocity demand value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Velocity: ");
  Serial.println(motor.getVelocityDemandValue());
  delay(1000);
  // ...
}
```

getTorqueDemandValue()

Description

Get the torque demand value object.

Relevant Objects

- Object 6074_h: Torque demand value

Syntax

```
int16_t getTorqueDemandValue();
```

Parameters

None.

Return Value

Return the torque demand value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();

  motor.enable();
}

void loop() {
  Serial.print("Torque: ");
  Serial.println(motor.getTorqueDemandValue());
  delay(1000);
  // ...
}
```

getDigitalInputs()

Description

Get the digital inputs object.

Relevant Objects

- Object $60FD_h$: Digital inputs

Syntax

```
uint32_t getDigitalInputs();
```

Parameters

None.

Return Value

Return the value of the digital inputs object:

Bit	Description
0	Negative limit switch
1	Positive limit switch
2	Home switch
3	Interlock
4 ... 15	Reserved
16 ... 31	Manufacturer Specific

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  master.start();
}

void loop() {
  Serial.print("Digital Inputs: ");
  Serial.println(motor.getDigitalInputs());
```

```
  delay(1000);
  // ...
}
```

2.4 Profile Position mode (pp) Related Functions

Profile Position mode (pp) related functions for the EthercatDevice_CiA402 class.

Functions:

- pp_SetVelocity()
- pp_SetAcceleration()
- pp_SetDeceleration()
- pp_SetMotionProfileType()
- pp_Run()
- pp_IsTargetReached()
- pp_CheckFollowingError()
- pp_Halt()
- pp_Resume()

pp_SetVelocity()

Description

Set the profile velocity object.

Relevant Objects

- Object 6081_h: Profile velocity

Syntax

int pp_SetVelocity(uint32_t value);

Parameters

- `[in] value`
  The profile velocity to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
```

```
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(30000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_SetAcceleration()

Description

Set the profile acceleration object.

Relevant Objects

- Object 6083_h: Profile acceleration

Syntax

```
int pp_SetAcceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile acceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
```

```
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(30000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_SetDeceleration()

Description

Set the profile deceleration object.

Relevant Objects

- Object 6084_h: Profile deceleration

Syntax

```
int pp_SetDeceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile deceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
```

```
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(30000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_SetMotionProfileType()

Description

Set the motion profile type object.

Relevant Objects

- Object 6086_h: Motion profile type

Syntax

```
int pp_SetMotionProfileType(int16_t value);
```

Parameters

- `[in] value`

  The motion profile type to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Linear ramp (trapezoidal profile)
1	$\sin^2$ ramp
2	Jerk-free ramp
3	Jerk-limited ramp
4 .. 32767	Reserved

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
```

```
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);

  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(30000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_Run()

Description

Move to the target position in Profile Position mode (pp).

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword
- Object $607A_h$: Target position

Syntax

```
int pp_Run(int32_t position, int relative = CIA402_PP_ABSOLUTE, bool
change_immediately = false);
```

Parameters

- `[in] position`
  The target position to be set.
- `[in] relative`
  A value indicating whether the position parameter is relative or absolute:

Definition	Code	Description
CIA402_PP_ABSOLUTE	0	position is an absolute value.
CIA402_PP_RELATIVE	1	position is a relative value.

- `[in] change_immediately`
  A Boolean to determine if the target position change should be executed instantly.
  - true: Interrupt the actual positioning and start the next positioning.
  - false: Finish the actual positioning and then start the next positioning.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();
```

```
  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);

  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(30000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_IsTargetReached()

Description

Check if the target position has been reached in Profile Position mode (pp).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int pp_IsTargetReached();
```

Parameters

None.

Return Value

Return whether the target position has been reached.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_CheckFollowingError()

Description

Check if the following error occurs in Profile Position mode (pp).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int pp_CheckFollowingError();
```

Parameters

None.

Return Value

Return whether a following error has occurred.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
  if (motor.pp_CheckFollowingError())
    Serial.println("Following Error.");

  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
```

```
  if (motor.pp_CheckFollowingError())
    Serial.println("Following Error.");
}
```

pp_Halt()

Description

Pause the current operation and wait for the drive to stop based on the configuration specified in `setHaltOptionCode()`.

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int pp_Halt(uint32_t timeout_ms = 10000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(1000000, CIA402_PP_RELATIVE, true);
  delay(3000);
```

```
  motor.pp_Halt();
  delay(3000);
  motor.pp_Resume();
  while (motor.pp_IsTargetReached() == 0);

  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-1000000, CIA402_PP_RELATIVE, true);
  delay(3000);
  motor.pp_Halt();
  delay(3000);
  motor.pp_Resume();
  while (motor.pp_IsTargetReached() == 0);
}
```

pp_Resume()

Description

Resume the paused operation that was called before pp_Halt().

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int pp_Resume();
```

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(1000000, CIA402_PP_RELATIVE, true);
  delay(3000);
  motor.pp_Halt();
  delay(3000);
```

```
  motor.pp_Resume();
  while (motor.pp_IsTargetReached() == 0);

  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
  motor.pp_Run(-1000000, CIA402_PP_RELATIVE, true);
  delay(3000);
  motor.pp_Halt();
  delay(3000);
  motor.pp_Resume();
  while (motor.pp_IsTargetReached() == 0);
}
```

2.5 Profile Velocity mode (pv) Related Functions

Profile Velocity mode (pv) related functions for the EthercatDevice_CiA402 class.

Functions:

- pv_SetAcceleration()
- pv_SetDeceleration()
- pv_SetMotionProfileType()
- pv_Run()
- pv_IsTargetReached()
- pv_CheckZeroSpeed()
- pv_CheckMaxSlippageError()
- pv_Halt()
- pv_Resume()

pv_SetAcceleration()

Description

Set the profile acceleration object.

Relevant Objects

- Object 6083_h: Profile acceleration

Syntax

```
int pv_SetAcceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile acceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

pv_SetDeceleration()

Description

Set the profile deceleration object.

Relevant Objects

- Object 6084_h: Profile deceleration

Syntax

```
int pv_SetDeceleration(uint32_t value);
```

Parameters

- `[in] value`
  The profile deceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

pv_SetMotionProfileType()

Description

Set the motion profile type object.

Relevant Objects

- Object 6086_h: Motion profile type

Syntax

```
int pv_SetMotionProfileType(int16_t value);
```

Parameters

- `[in] value`

  The motion profile type to be set:

Value	Description
-32768 ... -1	Manufacturer Specific
0	Linear ramp (trapezoidal profile)
1	$\sin^2$ ramp
2	Jerk-free ramp
3	Jerk-limited ramp
4 .. 32767	Reserved

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
```

```
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);

  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

pv_Run()

Description

Move at a target velocity continuously in Profile Velocity mode (pv).

Relevant Objects

- Object 6041_h: Statusword
- Object $60FF_h$: Target velocity

Syntax

```
int pv_Run(int32_t velocity);
```

Parameters

- `[in] velocity`
  The target velocity to be set.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

pv_IsTargetReached()

Description

Check if the target velocity has been reached in Profile Velocity mode (pv).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int pv_IsTargetReached();
```

Parameters

None.

Return Value

Return whether the target velocity has been reached.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);

  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
```

```
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

pv_CheckZeroSpeed()

Description

Check if the speed is zero in Profile Velocity mode (pv).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int pv_CheckZeroSpeed();
```

Parameters

None.

Return Value

Return whether the speed is zero.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);

  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
```

```
  motor.pv_Run(0);
  while (motor.pv_CheckZeroSpeed() == 0);
  delay(3000);
}
```

pv_CheckMaxSlippageError()

Description

Check if the maximum slippage error occurs in Profile Velocity mode (pv).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int pv_CheckMaxSlippageError();
```

Parameters

None.

Return Value

Return whether a maximum slippage error has occurred.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  if (motor.pv_CheckMaxSlippageError())
    Serial.println("Max skippage Error.");
```

```
  delay(3000);

  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  if (motor.pv_CheckMaxSlippageError())
    Serial.println("Max skippage Error.");
  delay(3000);
}
```

pv_Halt()

Description

Pause the current operation and wait for the drive to stop based on the configuration specified in `setHaltOptionCode()`.

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int pv_Halt(uint32_t timeout_ms = 10000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
}

void loop() {
  motor.pv_Halt();
  delay(1000);
```

```
  motor.pv_Resume();
  delay(1000);
}
```

pv_Resume()

Description

Resume the paused operation that was called before pv_Halt().

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int pv_Resume();
```

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
  motor.pv_Run(1000);
}

void loop() {
  motor.pv_Halt();
  delay(1000);
  motor.pv_Resume();
  delay(1000);
```

```
}
```

2.6 Profile Torque mode (tq) Related Functions

Profile Torque mode (tq) related functions for the EthercatDevice_CiA402 class.

Functions:

- tq_SetTorqueSlope()
- tq_SetTorqueProfileType()
- tq_SetMotorRatedCurrent()
- tq_SetMotorRatedTorque()
- tq_Run()
- tq_IsTargetReached()
- tq_Halt()
- tq_Resume()

tq_SetTorqueSlope()

Description

Set the torque slope object.

Relevant Objects

- Object 6087_h: Torque slope

Syntax

```
int tq_SetTorqueSlope(uint32_t value);
```

Parameters

- `[in] value`
  The torque slope to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_SetTorqueProfileType()

Description

Set the torque profile type object.

Relevant Objects

- Object 6088_h: Torque profile type

Syntax

```
int tq_SetTorqueProfileType(int16_t value);
```

Parameters

- `[in] value`
  The torque profile type to be set:

Value	Description
0000_h	Linear ramp (trapezoidal profile)
0001_h	$\sin^2$ ramp
0002_h .. $7FFF_h$	Reserved
8000_h .. $FFFF_h$	Manufacturer Specific

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
```

```
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);

  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_SetMotorRatedCurrent()

Description

Set the motor rated current object.

Relevant Objects

- Object 6075_h: Motor rated current

Syntax

```
int tq_SetMotorRatedCurrent(uint32_t value);
```

Parameters

- `[in] value`
  The motor rated current to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_SetMotorRatedTorque()

Description

Set the motor rated torque object.

Relevant Objects

- Object 6076_h: Motor rated torque

Syntax

```
int tq_SetMotorRatedTorque(uint32_t value);
```

Parameters

- `[in] value`
  The motor rated torque to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_Run()

Description

Drive continuously at the target torque in Profile Torque mode (tq).

Relevant Objects

- Object 6041_h: Statusword
- Object 6071_h: Target torque

Syntax

```
int tq_Run(int16_t torque);
```

Parameters

- `[in] torque`
  The target torque to be set.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
```

```
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_IsTargetReached()

Description

Check if the target torque has been reached in Profile Torque mode (tq).

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int tq_IsTargetReached();
```

Parameters

None.

Return Value

Return whether the target torque has been reached.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);

  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
```

```
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

tq_Halt()

Description

Pause the current operation and wait for the drive to stop based on the configuration specified in `setHaltOptionCode()`.

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int tq_Halt(uint32_t timeout_ms = 10000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
}

void loop() {
  motor.tq_Halt();
```

```
  delay(1000);
  motor.tq_Resume();
  delay(1000);
}
```

tq_Resume()

Description

Resume the paused operation that was called before tq_Halt().

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int tq_Resume();
```

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
  motor.tq_Run(50);
}

void loop() {
  motor.tq_Halt();
  delay(1000);
  motor.tq_Resume();
```

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

2.7 Homing mode (hm) Related Functions

Homing mode (hm) related functions for the EthercatDevice_CiA402 class.

Functions:

- hm_SetHomeOffset()
- hm_SetHomingMethod()
- hm_SetHomingSpeeds()
- hm_SetHomingAcceleration()
- hm_Run()
- hm_IsAttained()
- hm_Stop()

hm_SetHomeOffset()

Description

Set the home offset object.

Relevant Objects

- Object 607C$_h$: Home offset

Syntax

```
int hm_SetHomeOffset(int32_t value);
```

Parameters

- `[in] value`
  The home offset to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
}

void loop() {
```

```
  // put your main code here, to run repeatedly:

}
```

hm_SetHomingMethod()

Description

Set the homing method object.

Relevant Objects

- Object 6098_h: Homing method

Syntax

```
int hm_SetHomingMethod(int8_t value);
```

Parameters

- `[in] value`

  The homing method to be set:

Value	Description
-128 .. -1	Manufacturer Specific
0	No homing operation required
1..35	Methods 1 to 35
36 .. 127	Reserved

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
```

```
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
}

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

}
```

hm_SetHomingSpeeds()

Description

Set the homing speeds object.

Relevant Objects

- Object 6099_h: Homing speeds

Syntax

```
int hm_SetHomingSpeeds(uint32_t speed_during_search_switch, uint32_t
speed_during_search_zero);
```

Parameters

- `[in] speed_during_search_switch`
  The speed during search for switch to be set.
- `[in] speed_during_search_zero`
  The speed during search for zero to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
```

```
}

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

}
```

hm_SetHomingAcceleration()

Description

Set the homing acceleration object.

Relevant Objects

- Object $609A_h$: Homing acceleration

Syntax

```
int hm_SetHomingAcceleration(uint32_t value);
```

Parameters

- `[in] value`
  The homing acceleration to be set.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
}

void loop() {
```

```
  // put your main code here, to run repeatedly:

}
```

hm_Run()

Description

Initiate a homing operation in Homing mode (hm).

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int hm_Run();
```

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
}

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

```
}
```

hm_IsAttained()

Description

Check the status of the homing operation in Homing mode (hm). This status represents bits 12 and 13 of the controlword object.

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int hm_IsAttained();
```

Parameters

None.

Return Value

Return the status of the homing operation. When the status value equals CIA402_HM_RUNNING, it indicates that the homing process is still in progress. The bit definitions of the status are shown in the table below.

Bit	Name	Value	Description
0	CIA402_HM_ATTAINED	0	Homing mode not yet completed.
		1	Homing mode carried out successfully.
1	CIA402_HM_ERROR	0	No homing error.
		1	Homing error occurred.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  int status;

  Serial.begin(115200);
  while (!Serial);

  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();
```

```
  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while ((status = motor.hm_IsAttained()) == CIA402_HM_RUNNING);

  if (status & CIA402_HM_ATTAINED)
    Serial.println("Homing mode carried out successfully.");
  if (status & CIA402_HM_ERROR)
    Serial.println("Homing error occurred.");
}

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

}
```

hm_Stop()

Description

Stop the homing operation and wait for the drive to stop based on the configuration specified in setHaltOptionCode().

Relevant Objects

- Object 6040_h: Controlword
- Object 6041_h: Statusword

Syntax

```
int hm_Stop(uint32_t timeout_ms = 10000);
```

Parameters

- `[in] timeout_ms`
  Timeout in milliseconds.

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;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  delay(1000);
  motor.hm_Stop();
}
```

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

}
```

2.8 Function Group "Touch Probe" Related Functions

Touch Probe related functions for the EthercatDevice_CiA402 class.

Functions:

- enableTouchProbe1()
- enableTouchProbe2()
- disableTouchProbe1()
- disableTouchProbe2()
- isTouchProbe1ValueReady()
- isTouchProbe2ValueReady()
- readTouchProbe1Value()
- readTouchProbe2Value()

enableTouchProbe1()

Description

Enable the touch probe 1.

Relevant Objects

- Object $60B8_h$: Touch probe function
- Object $60B9_h$: Touch probe status
- Object $60D0_h$: Touch probe source

Syntax

```
int enableTouchProbe1(int trigger_mode, int trigger_src, int trigger_timing);
```

Parameters

- `[in] trigger_mode`
  Choose the trigger mode of touch probe 1:

Definition	Code	Description
CIA402_TPMODE_FIRST_EVENT	0	Trigger the first event.
CIA402_TPMODE_CONTINUOUS	1	Trigger events continuously.

- `[in] trigger_src`
  Choose the trigger source of touch probe 1:

Definition	Code	Description
CIA402_TPSRC_TP_INPUT	0x00010000	Trigger with touch probe input.
CIA402_TPSRC_ENCODER_Z	0x00020000	Trigger with zero impulse signal or position encoder.
CIA402_TPSRC_DIGITAL_INPUT1	1	Trigger with digital input 1.
CIA402_TPSRC_DIGITAL_INPUT2	2	Trigger with digital input 2.
CIA402_TPSRC_DIGITAL_INPUT3	3	Trigger with digital input 3.
CIA402_TPSRC_DIGITAL_INPUT4	4	Trigger with digital input 4.
CIA402_TPSRC_HW_ENCODER_Z	5	Trigger with digital input 5.
CIA402_TPSRC_SW_ENCODER_Z	6	Trigger with digital input 6.

- `[in] trigger_timing`
  Choose the trigger timing of touch probe 1:

Definition	Code	Description
CIA402_TPTIMING_NONE	0	Switch off all sampling of touch probe 1.
CIA402_TPTIMING_POS_EDGE	1	Enable sampling at positive edge of touch probe 1.
CIA402_TPTIMING_NEG_EDGE	2	Enable sampling at negative edge of touch probe 1
CIA402_TPTIMING_BOTH	3	Enable sampling at both positive and negative edges of touch probe 1 simultaneously.

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_CiA402 motor;

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

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

  motor.enableTouchProbe1(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe1ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 1 positive value: ");
  Serial.println(motor.readTouchProbe1Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe1();
}

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

}
```

enableTouchProbe2()

Description

Enable the touch probe 2.

Relevant Objects

- Object $60B8_h$: Touch probe function
- Object $60B9_h$: Touch probe status
- Object $60D0_h$: Touch probe source

Syntax

```
int enableTouchProbe2(int trigger_mode, int trigger_src, int trigger_timing);
```

Parameters

- `[in] trigger_mode`
  Choose the trigger mode of touch probe 2:

Definition	Code	Description
CIA402_TPMODE_FIRST_EVENT	0	Trigger the first event.
CIA402_TPMODE_CONTINUOUS	1	Trigger events continuously.

- `[in] trigger_src`
  Choose the trigger source of touch probe 2:

Definition	Code	Description
CIA402_TPSRC_TP_INPUT	0x00010000	Trigger with touch probe input.
CIA402_TPSRC_ENCODER_Z	0x00020000	Trigger with zero impulse signal or position encoder.
CIA402_TPSRC_DIGITAL_INPUT1	1	Trigger with digital input 1.
CIA402_TPSRC_DIGITAL_INPUT2	2	Trigger with digital input 2.
CIA402_TPSRC_DIGITAL_INPUT3	3	Trigger with digital input 3.
CIA402_TPSRC_DIGITAL_INPUT4	4	Trigger with digital input 4.
CIA402_TPSRC_HW_ENCODER_Z	5	Trigger with digital input 5.
CIA402_TPSRC_SW_ENCODER_Z	6	Trigger with digital input 6.

- [in] trigger_timing
  Choose the trigger timing of touch probe 2:

Definition	Code	Description
CIA402_TPTIMING_NONE	0	Switch off all sampling of touch probe 2.
CIA402_TPTIMING_POS_EDGE	1	Enable sampling at positive edge of touch probe 2.
CIA402_TPTIMING_NEG_EDGE	2	Enable sampling at negative edge of touch probe 2
CIA402_TPTIMING_BOTH	3	Enable sampling at both positive and negative edges of touch probe 2 simultaneously.

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_CiA402 motor;

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

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

  motor.enableTouchProbe2(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe2ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 2 positive value: ");
  Serial.println(motor.readTouchProbe2Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe2();
}

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

}
```

disableTouchProbe1()

Description

Disable the touch probe 1.

Relevant Objects

- Object $60B8_h$: Touch probe function
- Object $60B9_h$: Touch probe status

Syntax

```
int disableTouchProbe1();
```

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 blocking and cannot be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe1(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe1ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 1 positive value: ");
  Serial.println(motor.readTouchProbe1Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe1();
}

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

```
}
```

disableTouchProbe2()

Description

Disable the touch probe 2.

Relevant Objects

- Object $60B8_h$: Touch probe function
- Object $60B9_h$: Touch probe status

Syntax

```
int disableTouchProbe2();
```

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 blocking and cannot be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe2(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe2ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 2 positive value: ");
  Serial.println(motor.readTouchProbe2Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe2();
}

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

```
}
```

isTouchProbe1ValueReady()

Description

Check if a positive or negative edge has occurred on the touch probe 1 signal source.

Relevant Objects

- Object $60B9_h$: Touch probe status

Syntax

```
bool isTouchProbe1ValueReady(int value_selector);
```

Parameters

- `[in] value_selector`
  Choose the event to check on touch probe 1:

Definition	Code	Description
CIA402_TPVALUE_POS_EDGE	0	Check if a positive edge has occurred on the touch probe 1 signal source.
CIA402_TPVALUE_NEG_EDGE	1	Check if a negative edge has occurred on the touch probe 1 signal source.

Return Value

Return whether a positive or negative edge has occurred on the touch probe 1 signal source.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe1(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe1ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 1 positive value: ");
  Serial.println(motor.readTouchProbe1Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe1();
```

```
}

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

}
```

isTouchProbe2ValueReady()

Description

Check if a positive or negative edge has occurred on the touch probe 2 signal source.

Relevant Objects

- Object $60B9_h$: Touch probe status

Syntax

```
bool isTouchProbe2ValueReady(int value_selector);
```

Parameters

- `[in] value_selector`
  Choose the event to check on touch probe 2:

Definition	Code	Description
CIA402_TPVALUE_POS_EDGE	0	Check if a positive edge has occurred on the touch probe 2 signal source.
CIA402_TPVALUE_NEG_EDGE	1	Check if a negative edge has occurred on the touch probe 2 signal source.

Return Value

Return whether a positive or negative edge has occurred on the touch probe 2 signal source.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe2(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe2ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 2 positive value: ");
  Serial.println(motor.readTouchProbe2Value(CIA402_TPVALUE_POS_EDGE));
  motor.disableTouchProbe2();
```

```
}

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

}
```

readTouchProbe1Value()

Description

Read the touch probe position 1 positive/negative value object.

Relevant Objects

- Object $60BA_h$: Touch probe position 1 positive value
- Object $60BB_h$: Touch probe position 1 negative value

Syntax

```
int32_t readTouchProbe1Value(int value_selector);
```

Parameters

- `[in] value_selector`
  Choose the data source for reading the position value of touch probe 1:

Definition	Code	Description
CIA402_TPVALUE_POS_EDGE	0	Read the touch probe position 1 positive value. ($60BA_h$)
CIA402_TPVALUE_NEG_EDGE	1	Read the touch probe position 1 negative value. ($60BB_h$)

Return Value

Return the touch probe position 1 value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe1(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe1ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 1 positive value: ");
  Serial.println(motor.readTouchProbe1Value(CIA402_TPVALUE_POS_EDGE));
```

```
  motor.disableTouchProbe1();
}

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

}
```

readTouchProbe2Value()

Description

Read the touch probe position 2 positive/negative value object.

Relevant Objects

- Object $60BC_h$: Touch probe position 2 positive value
- Object $60BD_h$: Touch probe position 2 negative value

Syntax

```
int32_t readTouchProbe2Value(int value_selector);
```

Parameters

- `[in] value_selector`

  Choose the data source for reading the position value of touch probe 2:

Definition	Code	Description
CIA402_TPVALUE_POS_EDGE	0	Read the touch probe position 2 positive value. ($60BC_h$)
CIA402_TPVALUE_NEG_EDGE	1	Read the touch probe position 2 negative value. ($60BD_h$)

Return Value

Return the touch probe position 2 value.

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 if the related object is mapped to PDO.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

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

  motor.enableTouchProbe2(CIA402_TPMODE_FIRST_EVENT, CIA402_TPSRC_TP_INPUT,
CIA402_TPTIMING_POS_EDGE);
  while (motor.isTouchProbe2ValueReady(CIA402_TPVALUE_POS_EDGE) == false);
  Serial.print("Touch probe position 2 positive value: ");
  Serial.println(motor.readTouchProbe2Value(CIA402_TPVALUE_POS_EDGE));
```

```
  motor.disableTouchProbe2();
}

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

}
```

2.9 Low-level functions for mode-specific flow control

Low-level functions for mode-specific flow control related functions for the EthercatDevice_CiA402 class.

Functions:

- setHaltBit()
- isTargetReached()
- setModeSpecificBit4()
- setModeSpecificBit5()
- setModeSpecificBit6()
- checkModeSpecificBit12()
- checkModeSpecificBit13()

setHaltBit()

Description

Set the halt bit in the controlword object.

Relevant Objects

- Object 6040_h: Controlword

Syntax

```
int setHaltBit(bool halt);
```

Parameters

- `[in] halt`
  A Boolean value used to halt the servo drive.
  - true: Halt the servo drive.
  - false: Resume the servo drive.

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 non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();

  motor.setTargetVelocity(1000);
  delay(3000);
  motor.setHaltBit(true);
  delay(3000);
  motor.setHaltBit(false);
  delay(3000);
  motor.setTargetVelocity(0);
  delay(3000);
```

```
}

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

}
```

isTargetReached()

Description

Check if the target has been reached. It checks the target reached bit in the statusword object.

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int isTargetReached();
```

Parameters

None.

Return Value

Return whether the target has been reached.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);

  motor.setTargetPosition(-100000);
```

```
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
}
```

setModeSpecificBit4()

Description

Set the bit 4 in the controlword object.

Relevant Objects

- Object 6040_h: Controlword

Syntax

```
int setModeSpecificBit4(bool bit_value);
```

Parameters

- `[in] bit_value`
  A Boolean value used to set or clear the bit 4 in the controlword object.
  - true: Set the bit value to 1.
  - false: Clear the bit value to 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::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
```

```
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);

  motor.setTargetPosition(-100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
}
```

setModeSpecificBit5()

Description

Set the bit 5 in the controlword object.

Relevant Objects

- Object 6040_h: Controlword

Syntax

```
int setModeSpecificBit5(bool bit_value);
```

Parameters

- `[in] bit_value`
  A Boolean value used to set or clear the bit 5 in the controlword object.
  - true: Set the bit value to 1.
  - false: Clear the bit value to 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::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
```

```
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);

  motor.setTargetPosition(-100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
}
```

setModeSpecificBit6()

Description

Set the bit 6 in the controlword object.

Relevant Objects

- Object 6040_h: Controlword

Syntax

```
int setModeSpecificBit6(bool bit_value);
```

Parameters

- `[in] bit_value`
  A Boolean value used to set or clear the bit 6 in the controlword object.
  - true: Set the bit value to 1.
  - false: Clear the bit value to 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::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
```

```
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);

  motor.setTargetPosition(-100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
}
```

checkModeSpecificBit12()

Description

Check the value of bit 12 in the statusword object.

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int checkModeSpecificBit12();
```

Parameters

None.

Return Value

Return the value of bit 12 in the statusword.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);

  motor.setTargetPosition(-100000);
```

```
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
}
```

checkModeSpecificBit13()

Description

Check the value of bit 13 in the statusword object.

Relevant Objects

- Object 6041_h: Statusword

Syntax

```
int checkModeSpecificBit13();
```

Parameters

None.

Return Value

Return the value of bit 13 in the statusword.

Comment

This function must be called after a successful execution of `EthercatMaster::start()`. This function is non-blocking and can be called in callback functions.

Example Code

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

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

  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
}

void loop() {
  motor.setTargetPosition(100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
```

```
  while (motor.isTargetReached() == 0);
  if (motor.checkModeSpecificBit13())
    Serial.println("Following Error.");

  motor.setTargetPosition(-100000);
  motor.setModeSpecificBit6(true);
  motor.setModeSpecificBit5(true);
  motor.setModeSpecificBit4(true);
  while (motor.checkModeSpecificBit12() == 0);
  motor.setModeSpecificBit4(false);
  while (motor.checkModeSpecificBit12() == 1);
  while (motor.isTargetReached() == 0);
  if (motor.checkModeSpecificBit13())
    Serial.println("Following Error.");

}
```

Ch. 3

Example

3.1 Profile Position (pp) control

Implement position control on a CiA 402 EtherCAT SubDevice supporting Profile Position mode (pp).

- Move a relative distance of 10,000 units in the positive direction.
- Move a relative distance of 10,000 units in the negative direction.

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);
  master.start();

  motor.enable();
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
}

void loop() {
  motor.pp_Run(100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
  motor.pp_Run(-100000, CIA402_PP_RELATIVE, true);
  while (motor.pp_IsTargetReached() == 0);
}
```

3.2 Profile Position (pp) control in cyclic callback

Implement position control on a CiA 402 EtherCAT SubDevice supporting Profile Position mode (pp) in cyclic callback.

- Move a relative distance of 10,000 units in the positive direction.
- Move a relative distance of 10,000 units in the negative direction.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 607Ah: Target position
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 6064h: Position actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

#define STATE_SET_COMMAND          (0)
#define STATE_CHECK_ACK_SET        (1)
#define STATE_CHECK_ACK_CLEAR      (2)
#define STATE_WAIT_TARGET_REACHED   (3)
int state = STATE_SET_COMMAND;
int toggle = 0;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  switch (state) {
    case STATE_SET_COMMAND:
      toggle = !toggle;
      motor.setTargetPosition(100000 * toggle - 100000 * !toggle);
      motor.setModeSpecificBit6(true);
      motor.setModeSpecificBit5(false);
      motor.setModeSpecificBit4(true);
```

```
      state = STATE_CHECK_ACK_SET;
      break;
    case STATE_CHECK_ACK_SET:
      if (motor.checkModeSpecificBit12()) {
        motor.setModeSpecificBit4(false);
        state = STATE_CHECK_ACK_CLEAR;
      }
      break;
    case STATE_CHECK_ACK_CLEAR:
      if (motor.checkModeSpecificBit12() == 0)
        state = STATE_WAIT_TARGET_REACHED;
      break;
    case STATE_WAIT_TARGET_REACHED:
      if (motor.pp_IsTargetReached())
        state = STATE_SET_COMMAND;
      break;
  }
}

void setup() {
  master.begin();

  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PP_MODE);

  /* RxPDO mapping configuration. */
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x607A0020);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x60640020);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);
```

```
  master.attachCyclicCallback(MyCyclicCallback);
  master.start();

  motor.enable();
  motor.pp_SetMotionProfileType(0);
  motor.pp_SetVelocity(100000);
  motor.pp_SetAcceleration(5000);
  motor.pp_SetDeceleration(5000);
}

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

3.3 Profile Velocity (pv) control

Implement velocity control on a CiA 402 EtherCAT SubDevice supporting Profile Velocity mode (pv).

- Move in the positive direction at a speed of 1,000 units per second for 3 seconds.
- Move in the negative direction at a speed of 1,000 units per second for 3 seconds.

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_PV_MODE);
  master.start();

  motor.enable();
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
}

void loop() {
  motor.pv_Run(1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);

  motor.pv_Run(-1000);
  while (motor.pv_IsTargetReached() == 0);
  delay(3000);
}
```

3.4 Profile Velocity (pv) control in cyclic callback

Implement velocity control on a CiA 402 EtherCAT SubDevice supporting Profile Velocity mode (pv).

- Move in the positive direction at a speed of 1,000 units per second for 3 seconds.
- Move in the negative direction at a speed of 1,000 units per second for 3 seconds.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 60FFh: Target velocity
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 606Ch: Velocity actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

int toggle = 0;
int cycle_count = 3000;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  if (++cycle_count < 3000)
    return;

  cycle_count = 0;
  toggle = !toggle;
  motor.setTargetVelocity(1000 * toggle - 1000 * !toggle);
}

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

```
  motor.setCiA402Mode(CIA402_PV_MODE);

  /* RxPDO mapping configuration. */
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x60FF0020);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x606C0020);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);

  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000);

  motor.enable();
  motor.pv_SetMotionProfileType(0);
  motor.pv_SetAcceleration(5000);
  motor.pv_SetDeceleration(5000);
}

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

3.5 Profile Torque (tq) control

Implement torque control on a CiA 402 EtherCAT SubDevice supporting Profile Torque mode (tq).

- Maintain a positive torque of 50 units for 3 seconds.
- Maintain a negative torque of 50 units for 3 seconds.

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
  master.start();

  motor.enable();
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
}

void loop() {
  motor.tq_Run(50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);

  motor.tq_Run(-50);
  while (motor.tq_IsTargetReached() == 0);
  delay(3000);
}
```

3.6 Profile Torque (tq) control in cyclic callback

Implement torque control on a CiA 402 EtherCAT SubDevice supporting Profile Torque mode (tq).

- Maintain a positive torque of 50 units for 3 seconds.
- Maintain a negative torque of 50 units for 3 seconds.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 6071h: Target torque
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 6077h: Torque actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

int toggle = 0;
int cycle_count = 3000;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  if (++cycle_count < 3000)
    return;

  cycle_count = 0;
  toggle = !toggle;
  motor.setTargetTorque(50 * toggle - 50 * !toggle);
}

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_TQ_MODE);
```

```
  /* RxPDO mapping configuration. */
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x60710010);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x60770010);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);

  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000);

  motor.enable();
  motor.tq_SetTorqueProfileType(0);
  motor.tq_SetTorqueSlope(200);
  motor.tq_SetMotorRatedTorque(0);
  motor.tq_SetMotorRatedCurrent(0);
}

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

3.7 Homing (hm) operation

Initiate the homing method 33 operation on a CiA 402 compliant EtherCAT SubDevice.

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

void setup() {
  master.begin();
  motor.attach(0, master);
  motor.setCiA402Mode(CIA402_HOMING_MODE);
  master.start();

  motor.enable();

  motor.hm_SetHomingMethod(CIA402_HM33);
  motor.hm_SetHomeOffset(0);
  motor.hm_SetHomingSpeeds(100, 20);
  motor.hm_SetHomingAcceleration(100);
  motor.hm_Run();
  while (motor.hm_IsAttained() == CIA402_HM_RUNNING);
}

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

3.8 Cyclic synchronous position (csp) control in cyclic callback

Implement Cyclic Synchronous Position (csp) control on a CiA 402 EtherCAT SubDevice.

- The target position is incremented by 1,000 units in each cycle.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 607Ah: Target position
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 6064h: Position actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

int32_t position = 0;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  motor.setTargetPosition(position += 1000);
}

void setup() {
  master.begin();

  motor.attach(0, master);
  motor.setDc(1000000);
  motor.setCiA402Mode(CIA402_CSP_MODE);

  /* RxPDO mapping configuration. */
```

```
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x607A0020);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x60640020);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);

  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000, ECAT_SYNC);

  motor.setTargetPosition(position = motor.getPositionActualValue());
  motor.enable();
}

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

3.9 Cyclic synchronous velocity (csv) control in cyclic callback

Implement Cyclic Synchronous Velocity (csv) control on a CiA 402 EtherCAT SubDevice.

- The target velocity is incremented by 1 unit each cycle until it reaches 3,000 units.
- The target velocity is decremented by 1 unit each cycle until it reaches -3,000 units.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- **Output PDO (RxPDO)**
  - Object 6040h: Controlword
  - Object 60FFh: Target velocity
- **Input PDO (TxPDO)**
  - Object 6041h: Statusword
  - Object 606Ch: Velocity actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

int32_t velocity = 0;
int toggle = 0;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  if (abs(velocity) >= 3000)
    toggle = !toggle;

  velocity = velocity +  toggle - !toggle;
  motor.setTargetVelocity(velocity);
}

void setup() {
  master.begin();
```

```
  motor.attach(0, master);
  motor.setDc(1000000);
  motor.setCiA402Mode(CIA402_CSV_MODE);

  /* RxPDO mapping configuration. */
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x60FF0020);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x606C0020);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);

  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000, ECAT_SYNC);

  motor.setTargetVelocity(0);
  motor.enable();
}

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

3.10 Cyclic synchronous torque (cst) control in cyclic callback

Implement Cyclic Synchronous Torque (cst) control on a CiA 402 EtherCAT SubDevice.

- The target torque is incremented by 1 unit each cycle until it reaches 50 units.
- The target torque is decremented by 1 unit each cycle until it reaches -50 units.

To operate in cyclic callback, the relevant objects must be mapped to PDOs as follows:

- Output PDO (RxPDO)
  - Object 6040h: Controlword
  - Object 6071h: Target torque
- Input PDO (TxPDO)
  - Object 6041h: Statusword
  - Object 6077h: Torque actual value

Here is the example code.

```
#include "Ethercat.h"

EthercatMaster master;
EthercatDevice_CiA402 motor;

int16_t torque = 0;
int toggle = 0;

void MyCyclicCallback()
{
  if (motor.getCiA402State() != CIA402_OPERATION_ENABLED)
    return;

  if (abs(torque) >= 50)
    toggle = !toggle;

  torque = torque +  toggle - !toggle;
  motor.setTargetTorque(torque);
}

void setup() {
  master.begin();
```

```
  motor.attach(0, master);
  motor.setDc(1000000);
  motor.setCiA402Mode(CIA402_CST_MODE);

  /* RxPDO mapping configuration. */
  motor.sdoDownload8(0x1C12, 0x00, 0);
  motor.sdoDownload8(0x1601, 0x00, 0);
  motor.sdoDownload32(0x1601, 0x01, 0x60400010);
  motor.sdoDownload32(0x1601, 0x02, 0x60710010);
  motor.sdoDownload8(0x1601, 0x00, 2);
  motor.sdoDownload16(0x1C12, 0x01, 0x1601);
  motor.sdoDownload8(0x1C12, 0x00, 1);
  /* TxPDO mapping configuration. */
  motor.sdoDownload8(0x1C13, 0x00, 0);
  motor.sdoDownload8(0x1A01, 0x00, 0);
  motor.sdoDownload32(0x1A01, 0x01, 0x60410010);
  motor.sdoDownload32(0x1A01, 0x02, 0x60770010);
  motor.sdoDownload8(0x1A01, 0x00, 2);
  motor.sdoDownload16(0x1C13, 0x01, 0x1A01);
  motor.sdoDownload8(0x1C13, 0x00, 1);

  master.attachCyclicCallback(MyCyclicCallback);
  master.start(1000000, ECAT_SYNC);

  motor.setTargetTorque(0);
  motor.enable();
}

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

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
ECAT_SUCCESS	0
ECAT_ERR_MODULE_INIT_FAIL	-100
ECAT_ERR_MODULE_GET_VERSION_FAIL	-101
ECAT_ERR_MODULE_VERSION_MISMATCH	-102
ECAT_ERR_MODULE_GENERIC_TRANSFER_INIT_FAIL	-103
ECAT_ERR_MASTER_DOWNLOAD_SETTINGS_FAIL	-200
ECAT_ERR_MASTER_SET_DEVICE_SETTINGS_FAIL	-201
ECAT_ERR_MASTER_GET_GROUP_INFO_FAIL	-202
ECAT_ERR_MASTER_GET_MASTER_INFO_FAIL	-203
ECAT_ERR_MASTER_GET_DEVICE_INFO_FAIL	-204
ECAT_ERR_MASTER_SET_GROUP_SETTINGS_FAIL	-205
ECAT_ERR_MASTER_MAPPING_INIT_FAIL	-206
ECAT_ERR_MASTER_INTERRUPT_INIT_FAIL	-207
ECAT_ERR_MASTER_ACTIVE_FAIL	-208
ECAT_ERR_MASTER_ENI_INITCMDS_FAIL	-209
ECAT_ERR_MASTER_NO_DEVICE	-210
ECAT_ERR_MASTER_ACYCLIC_INIT_FAIL	-300
ECAT_ERR_MASTER_ACYCLIC_REQUEST_FAIL	-301
ECAT_ERR_MASTER_ACYCLIC_BUSY	-302
ECAT_ERR_MASTER_ACYCLIC_TIMEOUT	-303
ECAT_ERR_MASTER_ACYCLIC_ERROR	-304
ECAT_ERR_MASTER_ACYCLIC_WRONG_STATUS	-405
ECAT_ERR_MASTER_GENERIC_SEND_FAIL	-400
ECAT_ERR_MASTER_GENERIC_RECV_FAIL	-401
ECAT_ERR_MASTER_NOT_BEGIN	-1000
ECAT_ERR_MASTER_WRONG_BUFFER_SIZE	-1001
ECAT_ERR_MASTER_REDUNDANCY_NO_DC	-1002
ECAT_ERR_MASTER_MEMORY_ALLOCATION_FAIL	-1003
ECAT_ERR_MASTER_OSLAYER_INIT_FAIL	-1004
ECAT_ERR_MASTER_NIC_INIT_FAIL	-1005
ECAT_ERR_MASTER_BASE_INIT_FAIL	-1006
ECAT_ERR_MASTER_CIA402_INIT_FAIL	-1007
ECAT_ERR_MASTER_SETUP_PDO_FAIL	-1008

ECAT_ERR_MASTER_SCAN_NETWORK_FAIL	-1009
ECAT_ERR_MASTER_START_MASTER_FAIL	-1010
ECAT_ERR_MASTER_CYCLETIME_TOO_SMALL	-1011
ECAT_ERR_MASTER_DUMP_OUTPUT_PDO_FAIL	-1012
ECAT_ERR_MASTER_CONFIG_DEVICE_FAIL	-1013
ECAT_ERR_MASTER_CONFIG_MAPPING_FAIL	-1014
ECAT_ERR_MASTER_WAIT_BUS_SYNC_TIMEOUT	-1015
ECAT_ERR_MASTER_WAIT_MASTER_SYNC_TIMEOUT	-1016
ECAT_ERR_MASTER_CYCLIC_START_FAIL	-1017
ECAT_ERR_MASTER_WRONG_BUFFER_POINTER	-1018
ECAT_ERR_MASTER_ENI_INIT_FAIL	-1050
ECAT_ERR_MASTER_ENI_MISMATCH	-1051
ECAT_ERR_MASTER_STOPPED	-1100
ECAT_ERR_MASTER_STARTED	-1101
ECAT_ERR_MASTER_NOT_IN_PREOP	-1102
ECAT_ERR_MASTER_NOT_IN_SAFEOP	-1103
ECAT_ERR_MASTER_NOT_IN_OP	-1104
ECAT_ERR_MASTER_II_TRANSITION_FAIL	-1200
ECAT_ERR_MASTER_IP_TRANSITION_FAIL	-1201
ECAT_ERR_MASTER_PS_TRANSITION_FAIL	-1202
ECAT_ERR_MASTER_SO_TRANSITION_FAIL	-1203
ECAT_ERR_DEVICE_NOT_EXIST	-2000
ECAT_ERR_DEVICE_NOT_ATTACH	-2001
ECAT_ERR_DEVICE_NO_MAILBOX	-2002
ECAT_ERR_DEVICE_NO_DC	-2003
ECAT_ERR_DEVICE_WRONG_INPUT	-2004
ECAT_ERR_DEVICE_MEMORY_ALLOCATION_FAIL	-2005
ECAT_ERR_DEVICE_VENDOR_ID_MISMATCH	-2006
ECAT_ERR_DEVICE_PRODUCT_CODE_MISMATCH	-2007
ECAT_ERR_DEVICE_NO_SUCH_FUNCTION	-2008
ECAT_ERR_DEVICE_FUNCTION_NOT_INIT	-2009
ECAT_ERR_DEVICE_BUSY	-2010
ECAT_ERR_DEVICE_TIMEOUT	-2011
ECAT_ERR_DEVICE_NO_DATA	-2012
ECAT_ERR_DEVICE_SII_READ_FAIL	-2100
ECAT_ERR_DEVICE_SII_WRITE_FAIL	-2101
ECAT_ERR_DEVICE_PDO_NOT_EXIST	-2200
ECAT_ERR_DEVICE_PDO_OUT_OF_RANGE	-2201
ECAT_ERR_DEVICE_FOE_NOT_SUPPORT	-2300
ECAT_ERR_DEVICE_FOE_REQUEST_FAIL	-2310
ECAT_ERR_DEVICE_FOE_TIMEOUT	-2311

ECAT_ERR_DEVICE_FOE_ERROR	-2312
ECAT_ERR_DEVICE_FOE_BUFFER_TOO_SMALL	-2313
ECAT_ERR_DEVICE_FOE_READ_FAIL	-2314
ECAT_ERR_DEVICE_FOE_WRITE_FAIL	-2315
ECAT_ERR_DEVICE_COE_SDO_NOT_SUPPORT	-2400
ECAT_ERR_DEVICE_COE_SDO_INFO_NOT_SUPPORT	-2401
ECAT_ERR_DEVICE_COE_BUSY	-2410
ECAT_ERR_DEVICE_COE_REQUEST_FAIL	-2411
ECAT_ERR_DEVICE_COE_TIMEOUT	-2412
ECAT_ERR_DEVICE_COE_ERROR	-2413
ECAT_ERR_DEVICE_CIA402_NOT_EXIST	-2500
ECAT_ERR_DEVICE_CIA402_ADD_FAIL	-2501
ECAT_ERR_DEVICE_CIA402_TYPE_MISMATCH	-2502
ECAT_ERR_DEVICE_CIA402_NO_MODE_SUPPORT	-2503
ECAT_ERR_DEVICE_CIA402_WRONG_MODE	-2504
ECAT_ERR_DEVICE_CIA402_MODE_NOT_SUPPORT	-2505
ECAT_ERR_DEVICE_CIA402_CHANGE_WRONG_STATE	-2506
ECAT_ERR_DEVICE_CIA402_WRITE_OBJECT_FAIL	-2507
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE	-2580
ECAT_ERR_DEVICE_CIA402_NO_SUCH_TOUCH_PROBE_SOURCE	-2581
ECAT_ERR_DEVICE_EOE_NOT_SUPPORT	-2600
ECAT_ERR_DEVICE_EOE_NO_SUCH_PORT	-2601
ECAT_ERR_DEVICE_EOE_TOO_MUCH_CONTENT	-2602
ECAT_ERR_DEVICE_EOE_BUSY	-2610
ECAT_ERR_DEVICE_EOE_REQUEST_FAIL	-2611
ECAT_ERR_DEVICE_EOE_TIMEOUT	-2612
ECAT_ERR_GROUP_WRONG_INPUT	-3000
ECAT_ERR_GROUP_NOT_ATTACH	-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 MDevice 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 SubDevice 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 MDevice 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 SubDevice 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 MDevice 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
Ayclic 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 MDevice 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 MDevice command interface failed.
Corrective Actions
Please contact the manufacturer.

-1007: ECAT_ERR_MASTER_CIA402_INIT_FAIL

Description
Initialization of EtherCAT MDevice 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 persist, 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 MDevice.
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 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 SubDevice.
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 Cyclic Callback.
Corrective Actions
Please avoid using SDO Upload/Download to access CiA 402 objects in Cyclic Callback. 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 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, SI0 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 can not 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 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, SI0 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.4 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:

Index (hex)	Object Type	Name
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

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.