EC Motion

Motion Control Library in “C++”

Controlling Drives based on CiA 402 device profile

acontis technologies

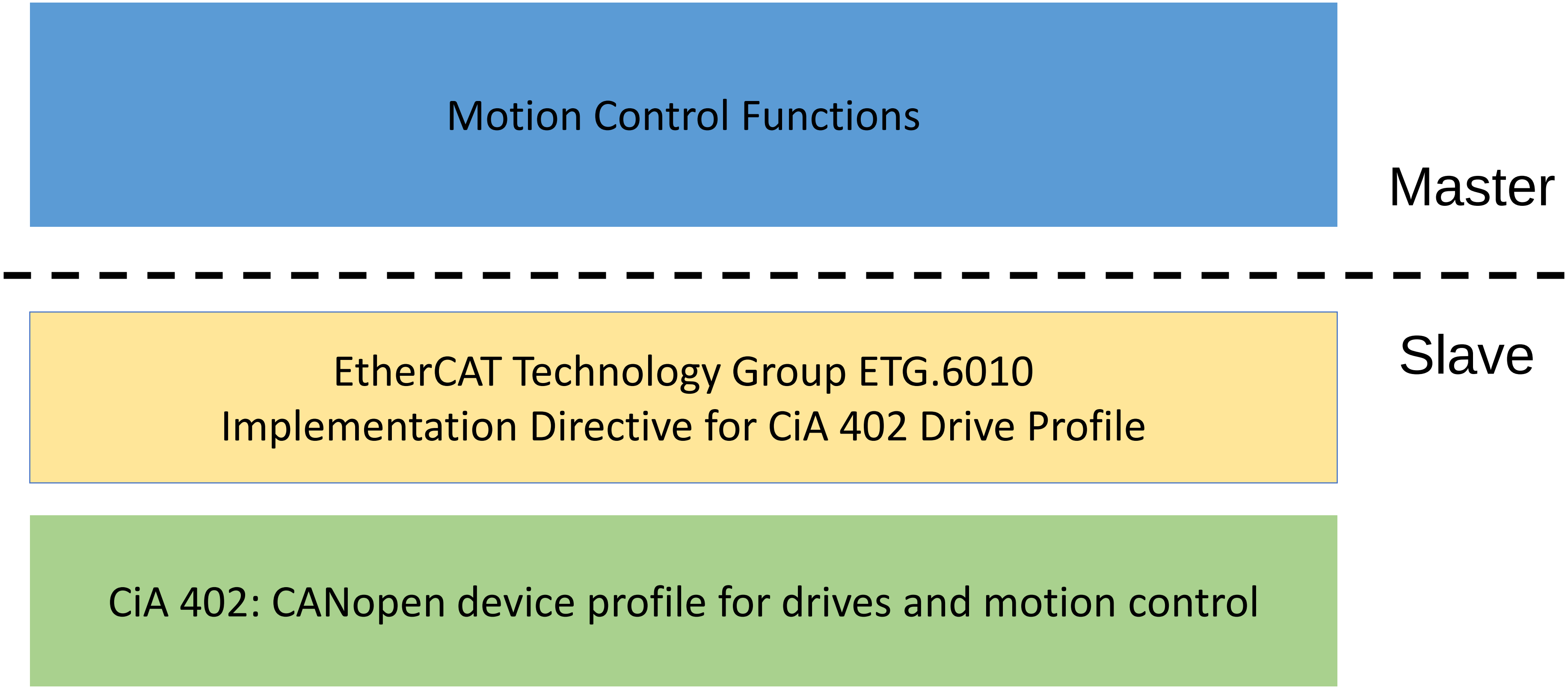
Content

- Introduction
- EC-Motion Library Architecture
- Administrative Functions Blocks
- Single Axis Motion Functions Blocks
- Examples
- Highlights

Introduction

- Most available drives with EtherCAT slave interface are based on the CiA 402 standard, e. g., Yaskawa, Copley, Beckhoff, Omron, …
- CiA 402 organizes parameters in a so called object dictionary and a drive state machine
- Based on this definitions it shall be possible to run drives from different manufacturers with the same application
- EtherCAT Technology Group (ETG) document ETG6010_V1i0i0_D_R_CiA402_ImplDirective gives additional implementation hints for using CiA 402 with EtherCAT

Introduction Standards


Motion Control Functions

Master

Slave

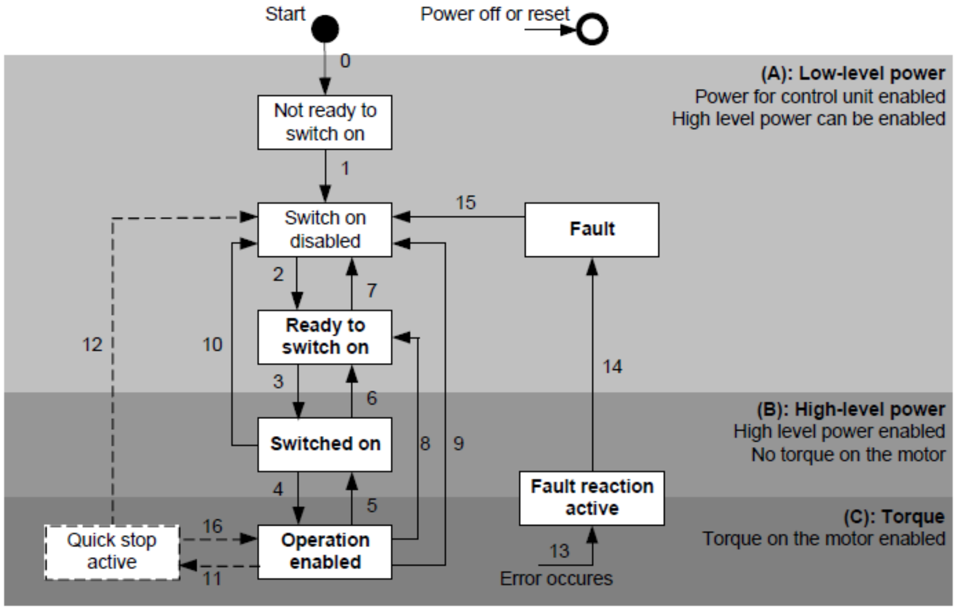
EtherCAT Technology Group ETG.6010
Implementation Directive for CiA 402 Drive Profile

CiA 402: CANopen device profile for drives and motion control

Introduction
CANopen device profile for drives and motion control

CiA 402: CANopen device profile for drives and motion control

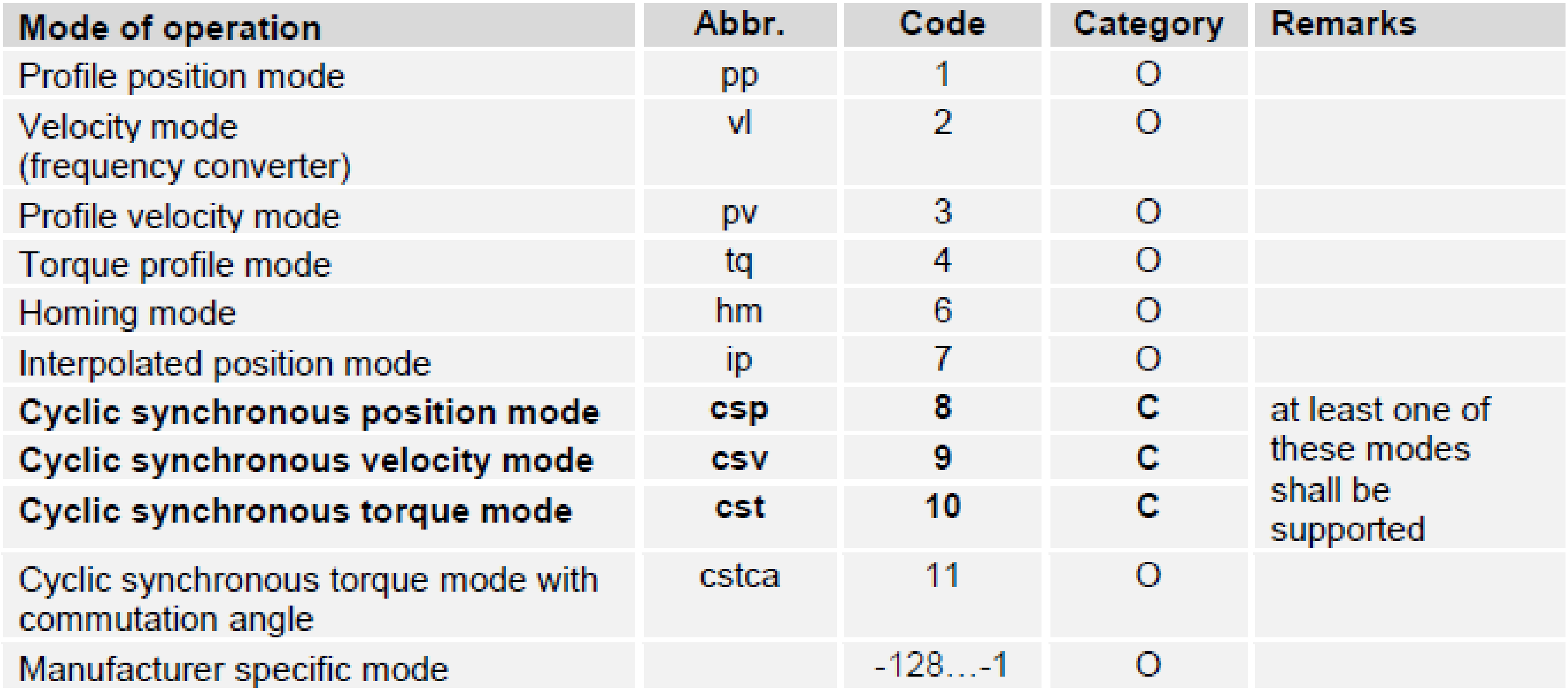
- CiA 402 organizes parameters in a so called object dictionary. Each parameter has a defined number (index + subindex) and meaning
  - Object 0x6040: Control Word
  - Object 0x6041: Status Word
  - Object 0x607A: Target Position
  - Object 0x6064: Actual Position
  - ....
- CiA 402 drive state machine



Introduction
ETG Implementation Directive for CiA 402 Drive Profile

ETG Implementation Directive for CiA 402 Drive Profile

- The drive has to support at least on of the cyclic operation modes: CSP or CSV or CST



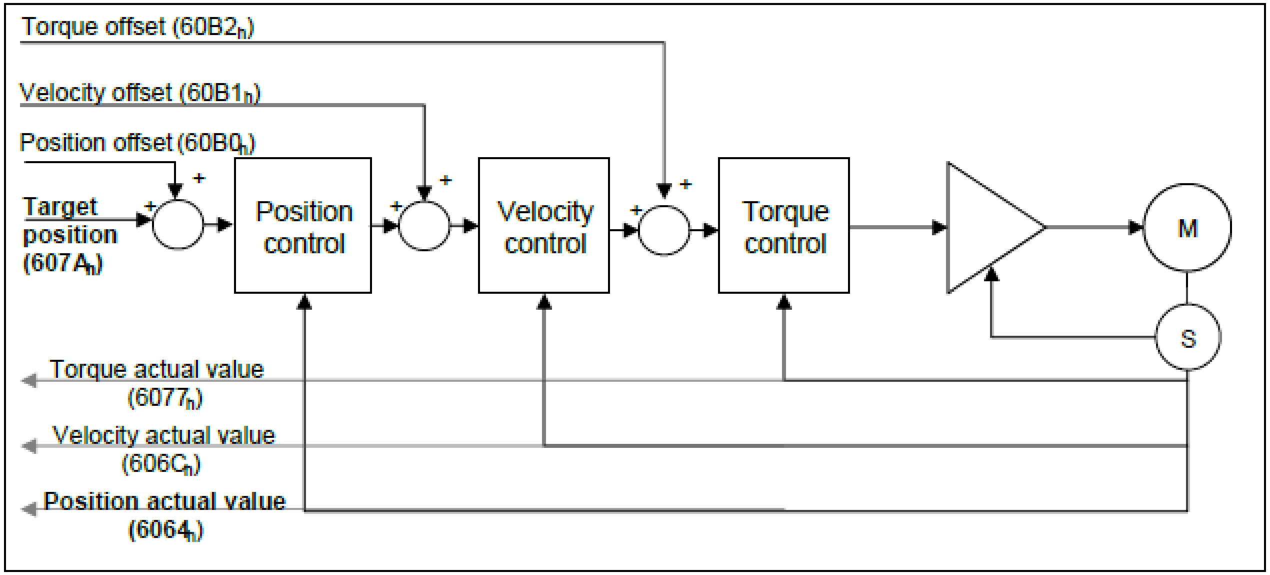
Mode of operation	Abbr.	Code	Category	Remarks
Profile position mode	pp	1	O	
Velocity mode (frequency converter)	vl	2	O	
Profile velocity mode	pv	3	O	
Torque profile mode	tq	4	O	
Homing mode	hm	6	O	
Interpolated position mode	ip	7	O	
Cyclic synchronous position mode	**csp**	**8**	**C**	at least one of these modes shall be supported
Cyclic synchronous velocity mode	**csv**	**9**	**C**	
Cyclic synchronous torque mode	**cst**	**10**	**C**	
Cyclic synchronous torque mode with commutation angle	cstca	11	O	
Manufacturer specific mode		-128…-1	O	

Introduction
ETG Implementation Directive for CiA 402 Drive Profile

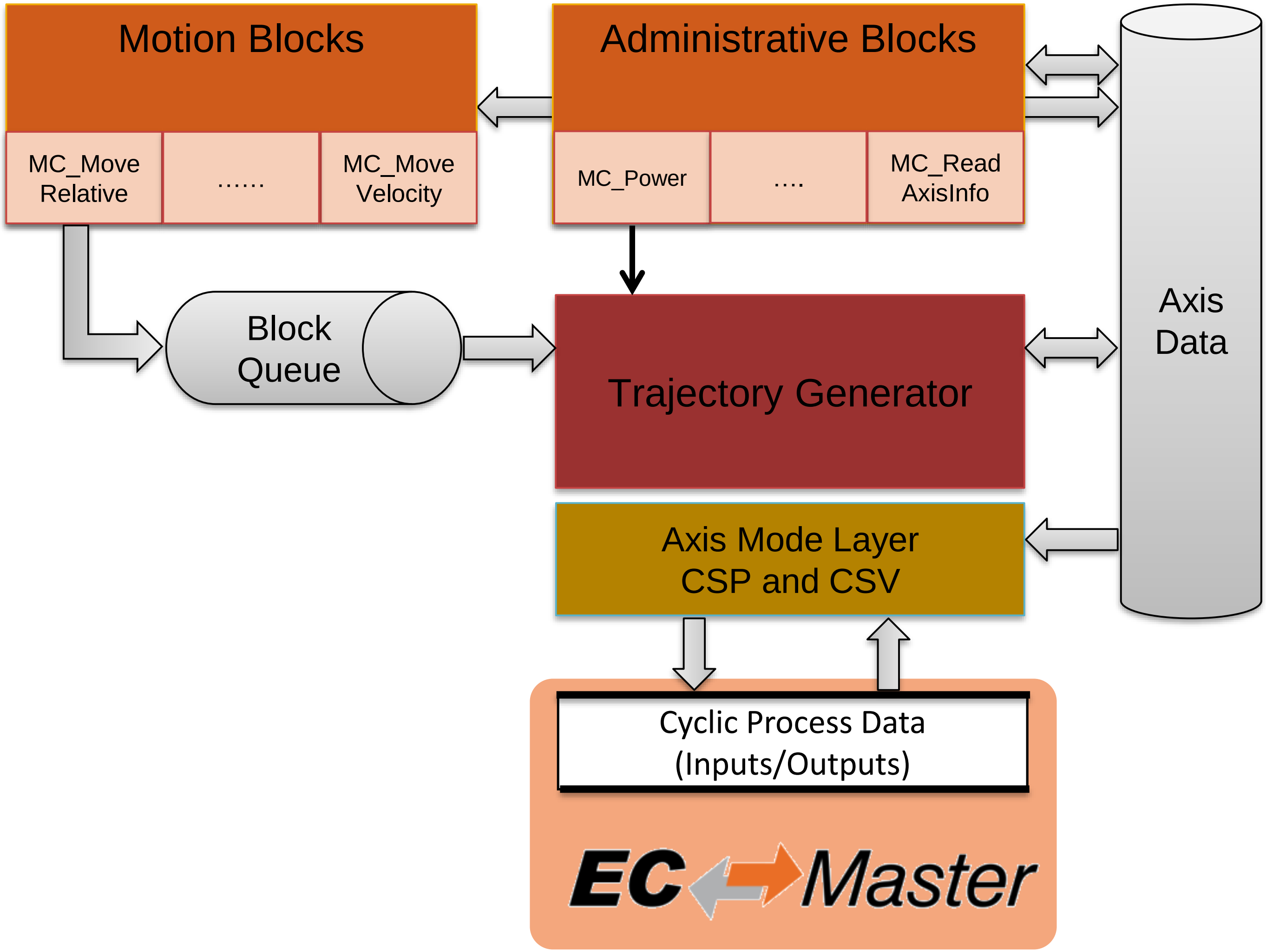
ETG Implementation Directive for CiA 402 Drive Profile

CSP: Cyclic Synchronous Position Mode

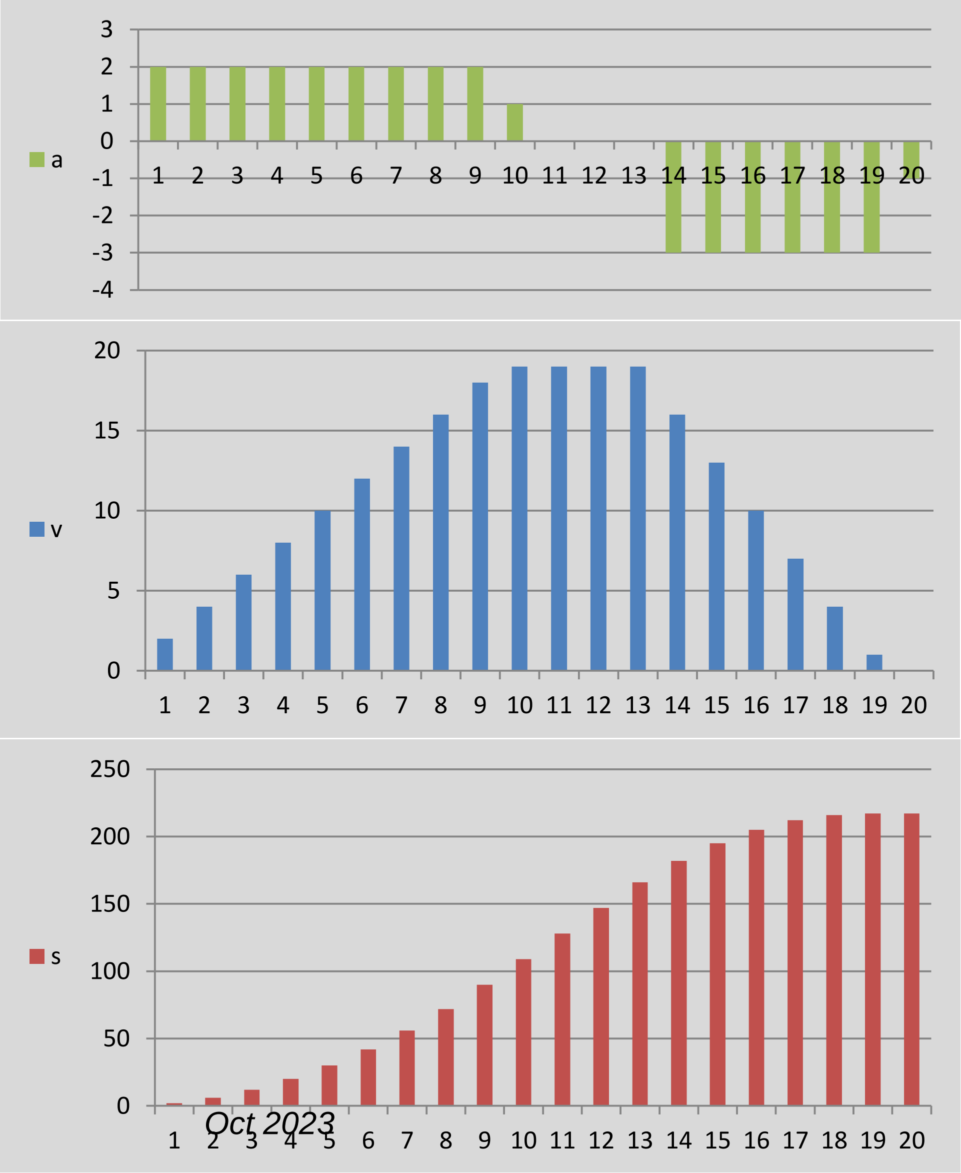
- Application has to set a new “Target position” in every cycle (trajectory generator)
- Position, Velocity and Torque are controlled by the drive



EC-Motion Control Library Architecture



Trajectory Generator
Example: CSP without jerk limitation



3 sections:

- Section 1: Accelerate
  a=2 up to v=29
- Section 2: Constant velocity
  v=29
- Section 3: Decelerate
  a=-3 down to v=0

EC-Motion Product Overview

- EC-Motion is a motion control solution for drives operating in a cyclic mode (CSP or CSV).
- EC-Motion is targeted to work in conjunction with the EC-Master (EtherCAT Master Stack). But EC-Master is not mandatory. Simulation only mode is supported as well.
- EC-Motion provides a Programmable Logic Controller (PLC) style interface. It is designed to be easy integrating in a PLC for controlling EtherCAT connected servo drives.
- The following EtherCAT drive profiles are supported:
  - CiA® 402: CANopen device profile for drives and motion control

Administrative Functions Blocks (1)

- **MC_POWER_T**: This Function Block controls the power stage (On or Off).
- **MC_HOME_T:** This Function Block commands the axis to perform the «search home» sequence.
- **MC_SETPOSITION_T:** This Function Block shifts the coordinate system
- **MC_READPARAMETER_T, MC_READBOOLPARAMETER_T**:
  Returns the value of a parameter
- **MC_WRITEPARAMETER_T, MC_WRITEBOOLPARAMETER_T**:
  Modifies the value of a parameter
- **MC_READDIGITALINPUT_T, MC_READDIGITALOUTPUT_T, MC_WRITEDIGITALOUTPUT_T:**
  Function Block gives access to the value of the input and outputs

Administrative Functions Blocks (2)

- **MC_READACTUALPOSITION_T**: This Function Block returns the actual position.
- **MC_READACTUALVELOCITY_T**: This Function Block returns the actual velocity.
- **MC_READMOTIONSTATE_T**: This Function Block returns the actual velocity.
- **MC_READAXISINFO_T:** This Function Block reads information concerning an axis
- **MC_READ_ERROR_T**: This Function Block presents general axis errors not relating to the Function Blocks
- **MC_RESET_T**: This Function Block makes the transition from the state ‘ErrorStop’ to ‘Standstill’ by resetting all internal axis-related errors

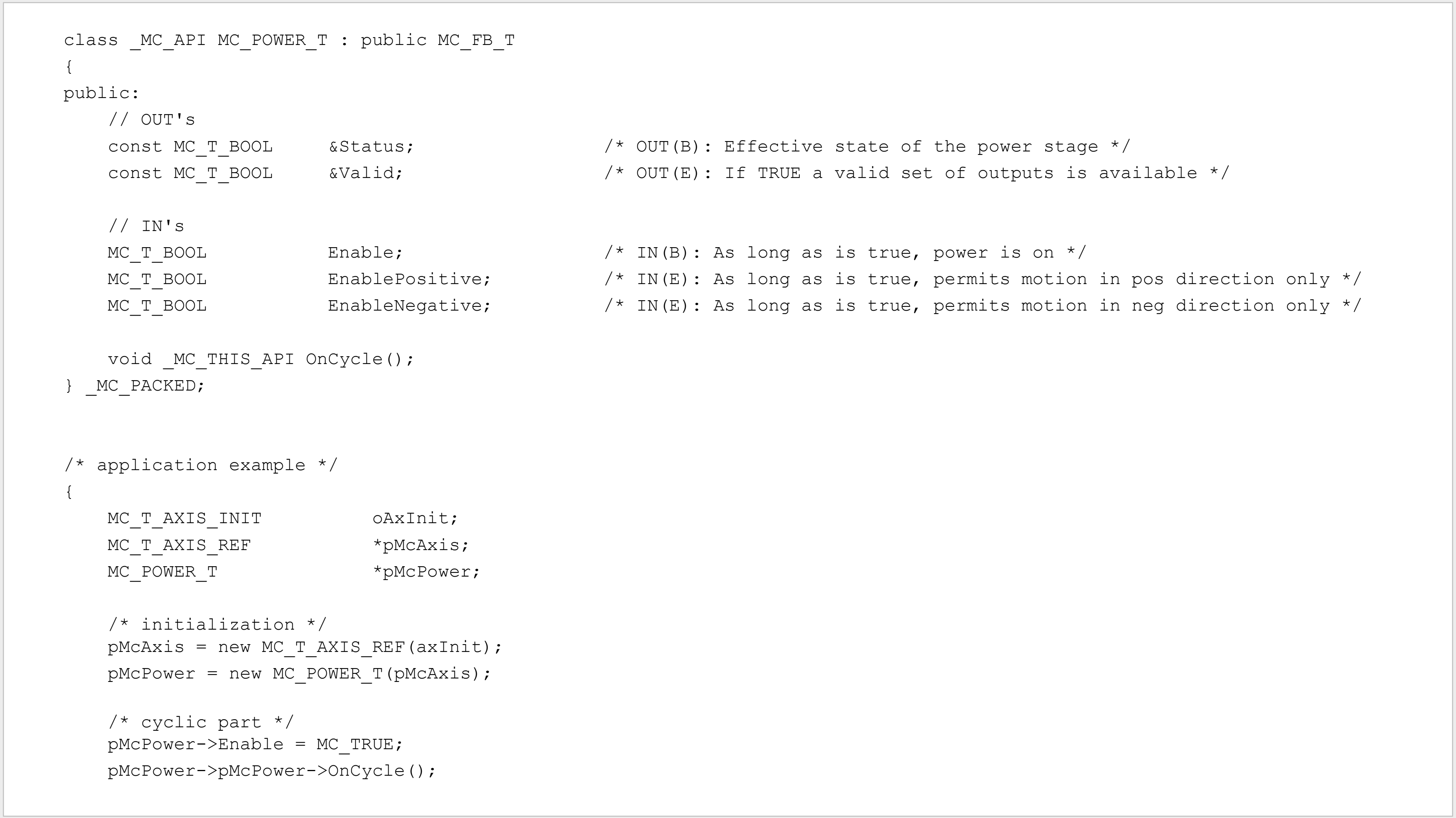
Single Axis Motion Functions Blocks (1)

- **MC_STOP_T**: Commands a controlled motion stop and transfers the axis to the state ‘Stopping’.
- **MC_HALT_T**: Commands a controlled motion stop and transfers the axis to the state ‘Standstill’.
- **MC_MOVEABSOLUTE_T**: Commands a controlled motion to a specified absolute position.
- **MC_MOVERELATIVE_T**: Commands a controlled motion of a specified distance relative to the set position at the time of the execution.
- **MC_MOVEVELOCITY_T**: Commands a never ending controlled motion at a specified velocity.
- **MC_MOVE_CONT_ABSOLUTE_T**: Commands a controlled motion to a specified absolute position ending with the specified velocity.
- **MC_MOVE_CONT_RELATIVE_T**: Commands a controlled motion of a specified relative distance ending with the specified velocity.
- **AMC_CHECK_TARGETPOS_REACHED_T**: Check if the actual position has reached the commanded position.

Single Axis Motion Functions Blocks (2)

- **MC_CalcMoveProfile, MC_CalcMoveProfileBuffered**: Calculate move times and segment distances for a specific movement without moving the axis.
- **MC_CalcMoveTimeAtPos**: Calculate time until a certain position is reached. MC
- **MC_DriveSetTargetStep**: Set velocity without using build-in trajectory generator.

Example: MC_Power in “C++” language



```
class _MC_API MC_POWER_T : public MC_FB_T
{
public:
    // OUT's
    const MC_T_BOOL     &Status;                /* OUT(B): Effective state of the power stage */
    const MC_T_BOOL     &Valid;                 /* OUT(E): If TRUE a valid set of outputs is available */

    // IN's
    MC_T_BOOL           Enable;                 /* IN(B): As long as is true, power is on */
    MC_T_BOOL           EnablePositive;         /* IN(E): As long as is true, permits motion in pos direction only */
    MC_T_BOOL           EnableNegative;         /* IN(E): As long as is true, permits motion in neg direction only */

    void _MC_THIS_API OnCycle();
} _MC_PACKED;


/* application example */
{
    MC_T_AXIS_INIT          oAxInit;
    MC_T_AXIS_REF           *pMcAxis;
    MC_POWER_T              *pMcPower;

    /* initialization */
    pMcAxis = new MC_T_AXIS_REF(axInit);
    pMcPower = new MC_POWER_T(pMcAxis);

    /* cyclic part */
    pMcPower->Enable = MC_TRUE;
    pMcPower->pMcPower->OnCycle();
```

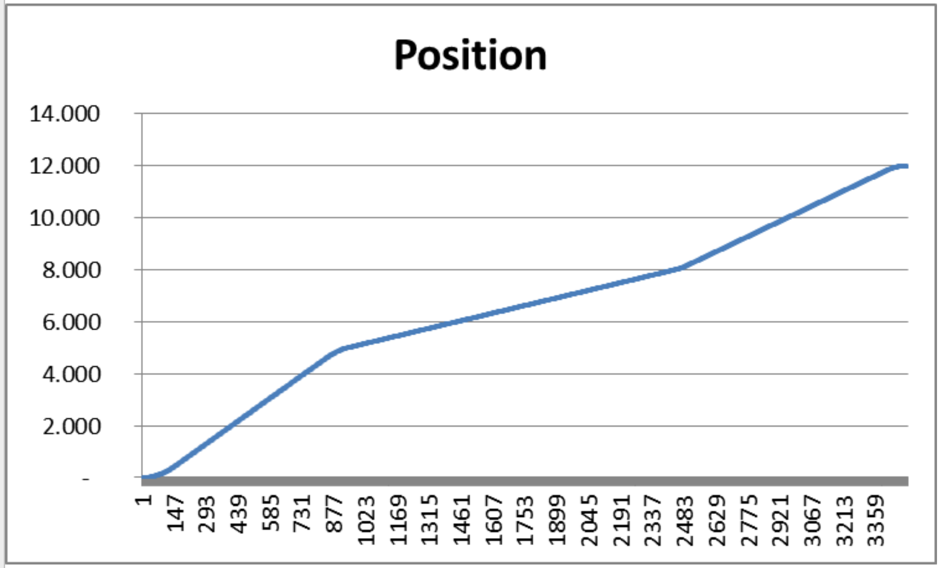
Example: MC_MoveRelative in “C++” language

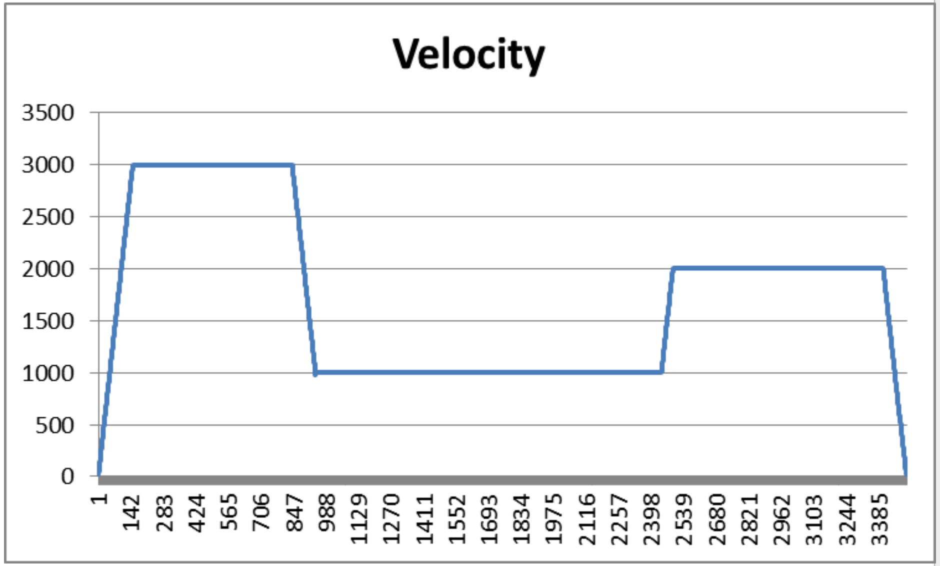
```
class _MC_API MC_MOVE_RELATIVE_T : public MC_BUFFERED_FB_T

{

public:

    // OUT's

    const MC_T_BOOL     &Done;                   /* OUT(B): The axis is within a range close to the target position */

    const MC_T_BOOL     &Busy;                   /* OUT(E): The FB is not finished and new output values are to be expected */


    // IN's

    MC_T_BOOL           Execute;                 /* IN(B): Start the motion at rising edge */

    MC_T_BOOL           ContinuousUpdate;        /* IN(E): Continuous Update (Trapezoid profile only) */

    MC_T_REAL           Distance;                /* IN(B): Relative distance for the motion */

    MC_T_REAL           Velocity;                /* IN(E): Value of the max velocity (always positive, not necessarily reached). */

    MC_T_REAL           Acceleration;            /* IN(E): Value of the acc (always positive, increasing energy of the motor). */

    MC_T_REAL           Deceleration;            /* IN(E): Value of the dec (always positive, decreasing energy of the motor). */

    MC_T_REAL           Jerk;                    /* IN(E): Value of the Jerk (always positive). */


    MC_MOVE_RELATIVE_T(MC_T_AXIS_REF *pAxis = MC_NULL);


    void _MC_THIS_API OnCycle();

}
```

Example MC_MoveRelative Buffermode = MC_BLENDING_LOW

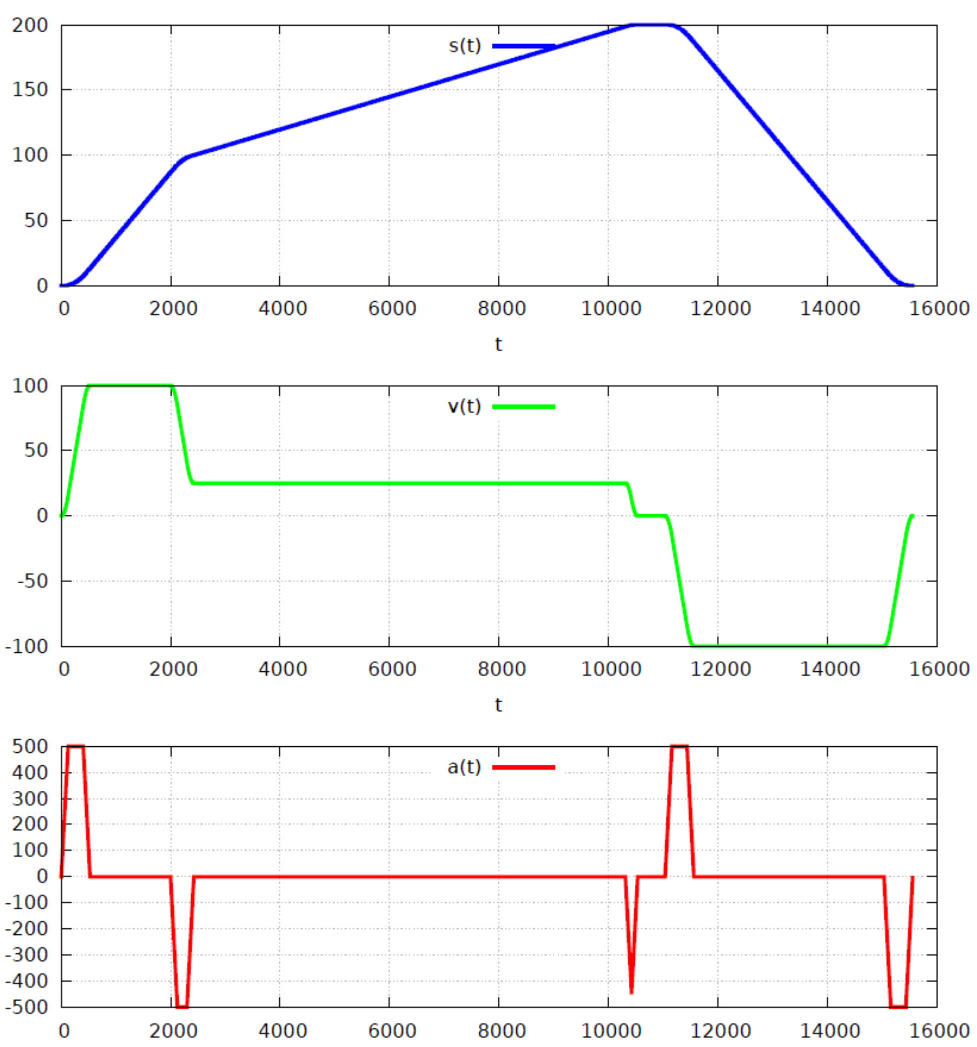
Sequence of three MC_MoveRelative without stop

- FB 1: MC_MoveRelative with Distance=5.0 and Velocity=3000
- FB 1: MC_MoveRelative with Distance=3.0 and Velocity=1000
- FB 1: MC_MoveRelative with Distance=4.0 and Velocity=2000



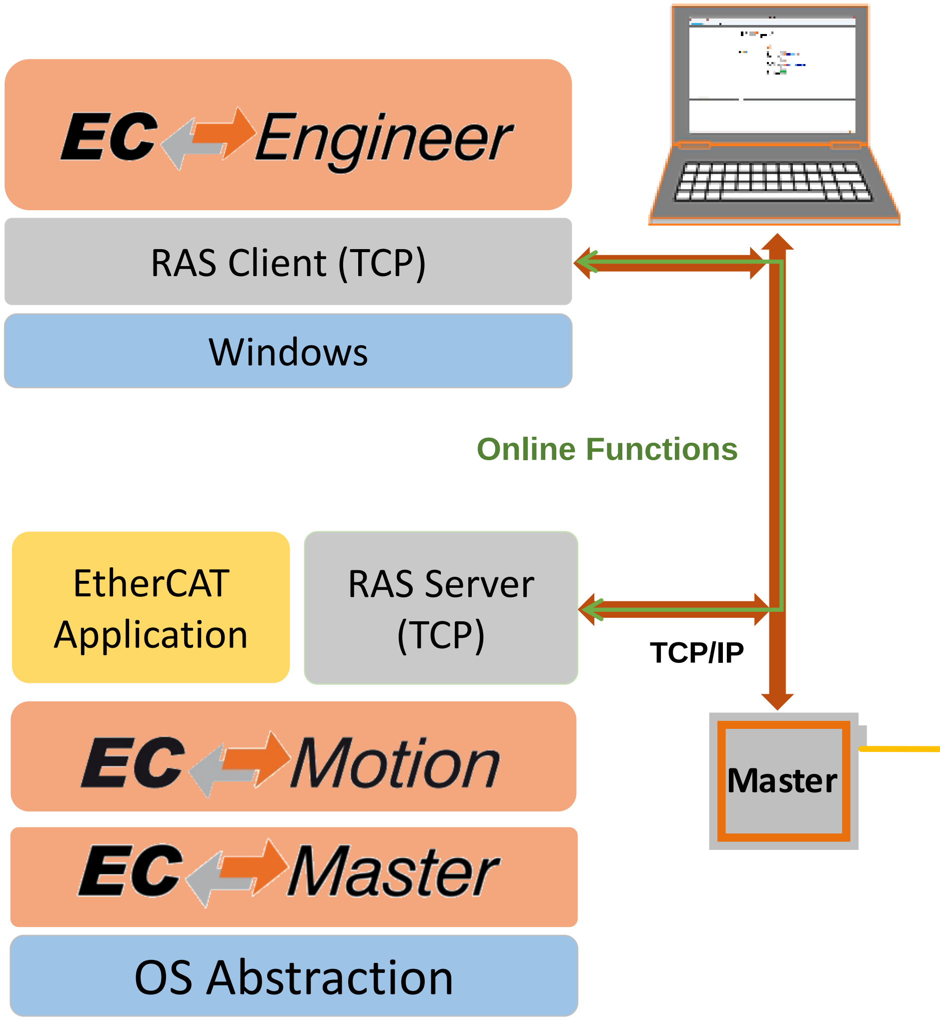


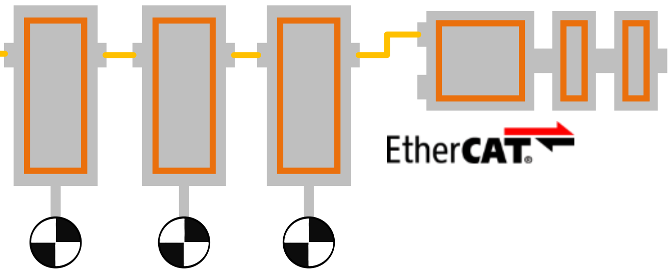
Drill example – jerk limited

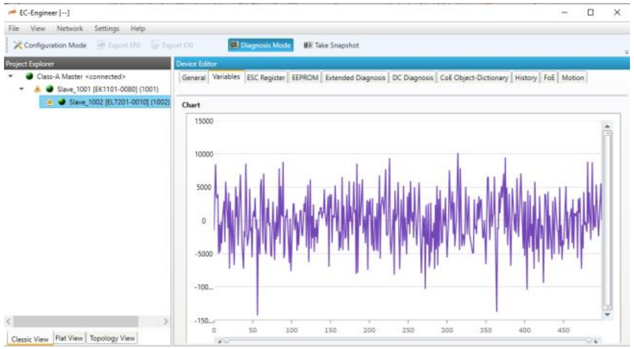


Diagnosis tools and
Example EcMasterDemoMotion

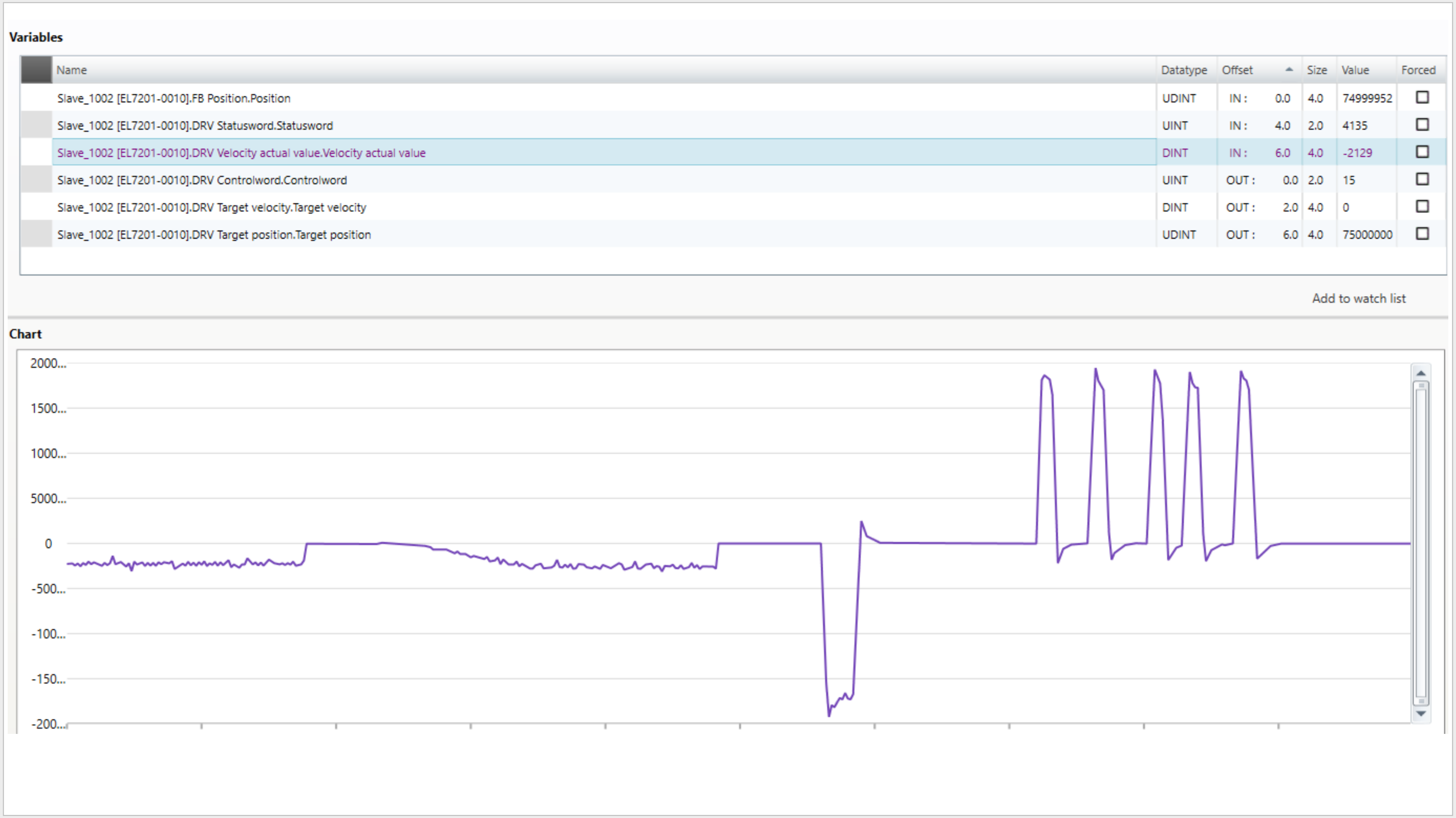
Diagnosis with EC-Engineer tool







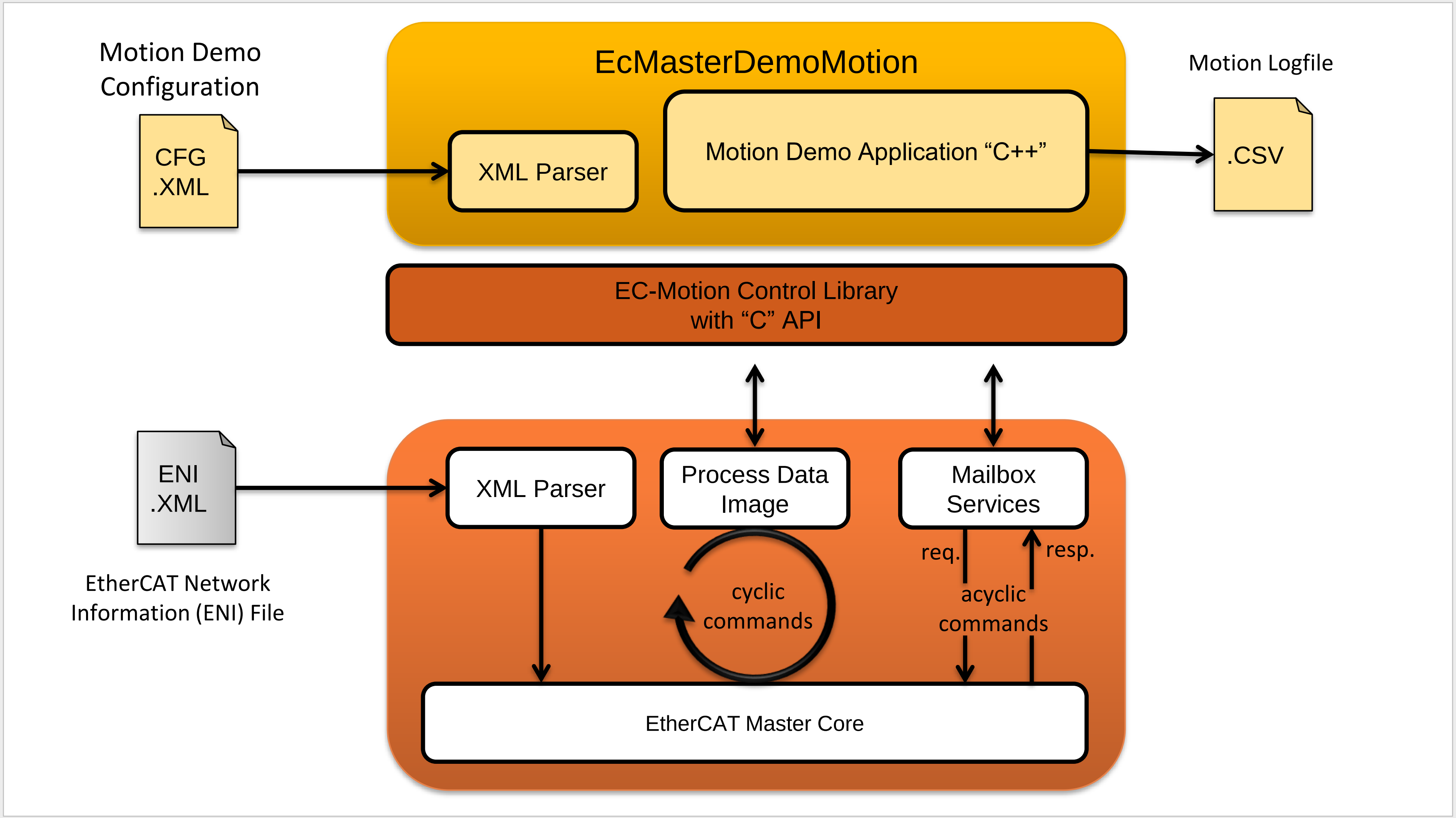
Diagnosis with EC-Engineer tool



Variables

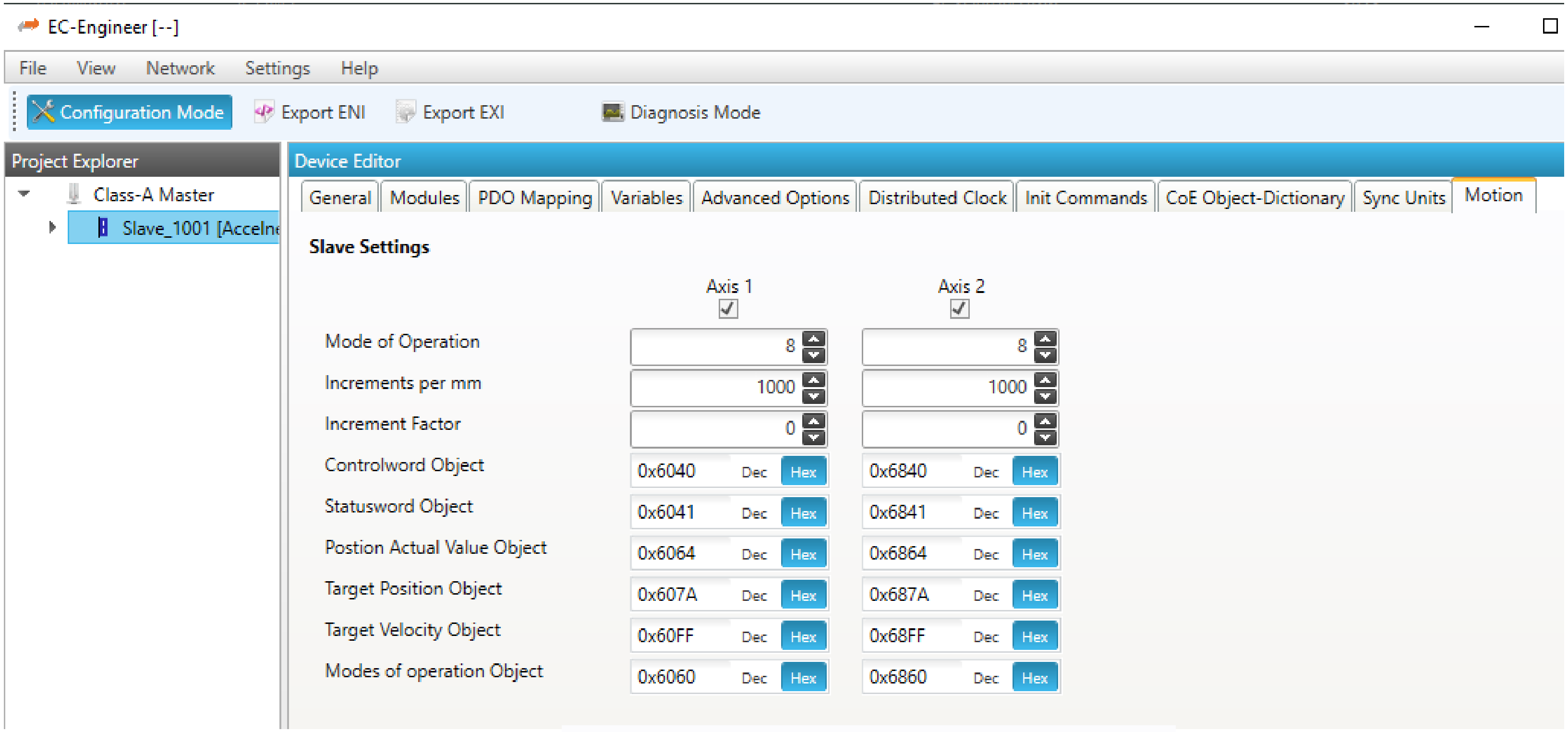
Name	Datatype	Offset	Size	Value	Forced
Slave_1002 [EL7201-0010].FB Position.Position	UDINT	IN : 0.0	4.0	74999952	□
Slave_1002 [EL7201-0010].DRV Statusword.Statusword	UINT	IN : 4.0	2.0	4135	□
Slave_1002 [EL7201-0010].DRV Velocity actual value.Velocity actual value	DINT	IN : 6.0	4.0	-2129	□
Slave_1002 [EL7201-0010].DRV Controlword.Controlword	UINT	OUT : 0.0	2.0	15	□
Slave_1002 [EL7201-0010].DRV Target velocity.Target velocity	DINT	OUT : 2.0	4.0	0	□
Slave_1002 [EL7201-0010].DRV Target position.Target position	UDINT	OUT : 6.0	4.0	75000000	□

Example EcMasterDemoMotion Software Architecture



Example EcMasterDemoMotion
Create demo configuration file in EC-Engineer

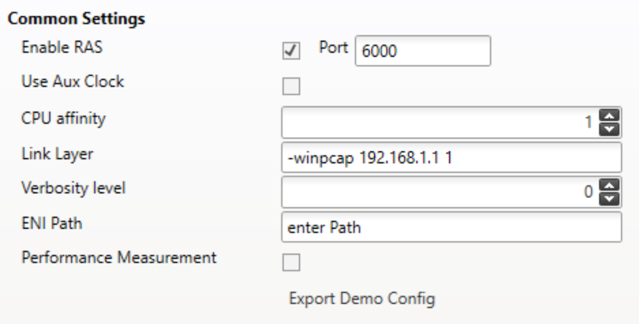
EC-Motion



EC-Engineer [--]

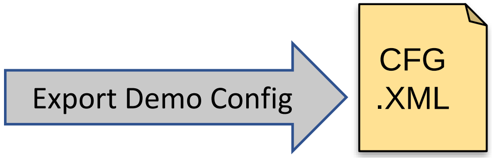
Slave Settings

	Axis 1 ☑	Axis 2 ☑
Mode of Operation	8	8
Increments per mm	1000	1000
Increment Factor	0	0
Controlword Object	0x6040 Dec Hex	0x6840 Dec Hex
Statusword Object	0x6041 Dec Hex	0x6841 Dec Hex
Postion Actual Value Object	0x6064 Dec Hex	0x6864 Dec Hex
Target Position Object	0x607A Dec Hex	0x687A Dec Hex
Target Velocity Object	0x60FF Dec Hex	0x68FF Dec Hex
Modes of operation Object	0x6060 Dec Hex	0x6860 Dec Hex

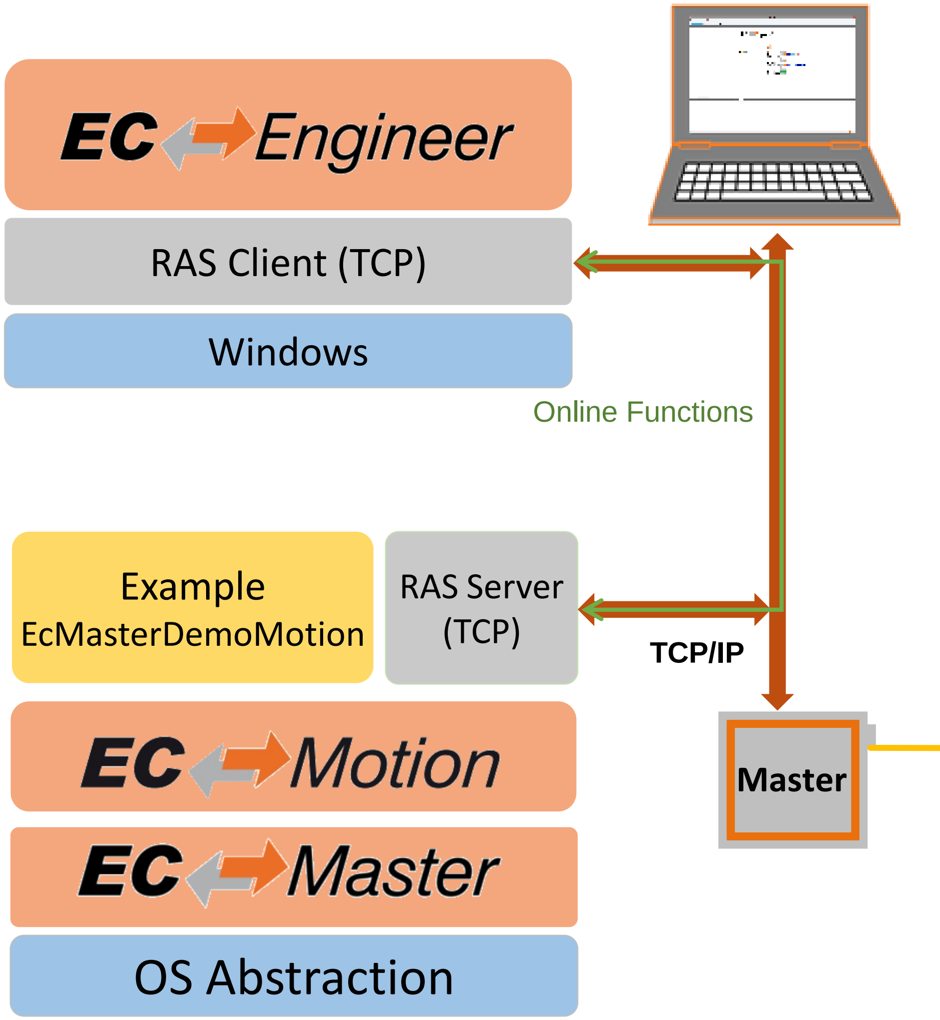


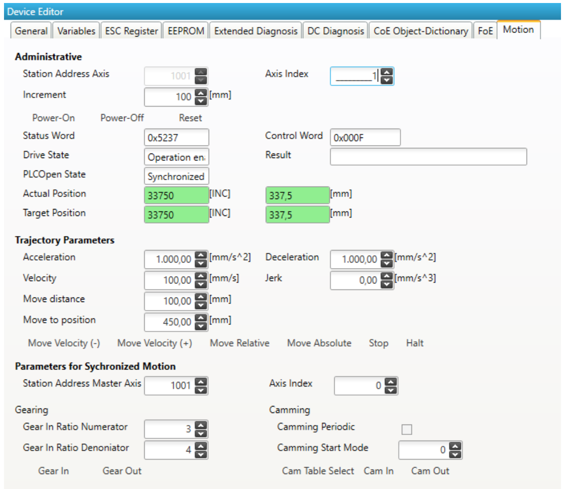
Common Settings

Enable RAS	☑ Port 6000
Use Aux Clock	☐
CPU affinity	1
Link Layer	-winpcap 192.168.1.1 1
Verbosity level	0
ENI Path	enter Path
Performance Measurement	☐



Example EcMasterDemoMotion Remote Control with EC-Engineer





Administrative

Station Address Axis: 1001
Axis Index: 1
Increment: 100 [mm]

Status Word: 0x5237
Control Word: 0x000F
Drive State: Operation en
Result:
PLCOpen State: Synchronized
Actual Position: 33750 [INC] 337,5 [mm]
Target Position: 33750 [INC] 337,5 [mm]

Trajectory Parameters

Acceleration: 1.000,00 [mm/s^2]
Deceleration: 1.000,00 [mm/s^2]
Velocity: 100,00 [mm/s]
Jerk: 0,00 [mm/s^3]
Move distance: 100,00 [mm]
Move to position: 450,00 [mm]

Parameters for Sychronized Motion

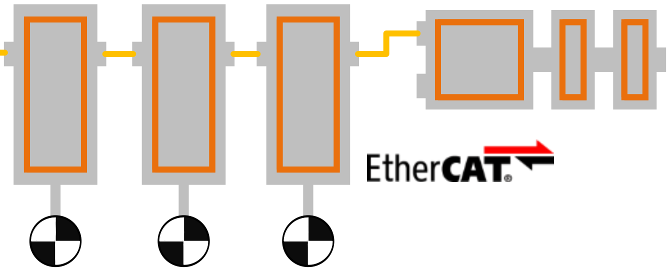
Station Address Master Axis: 1001
Axis Index: 0

Gearing

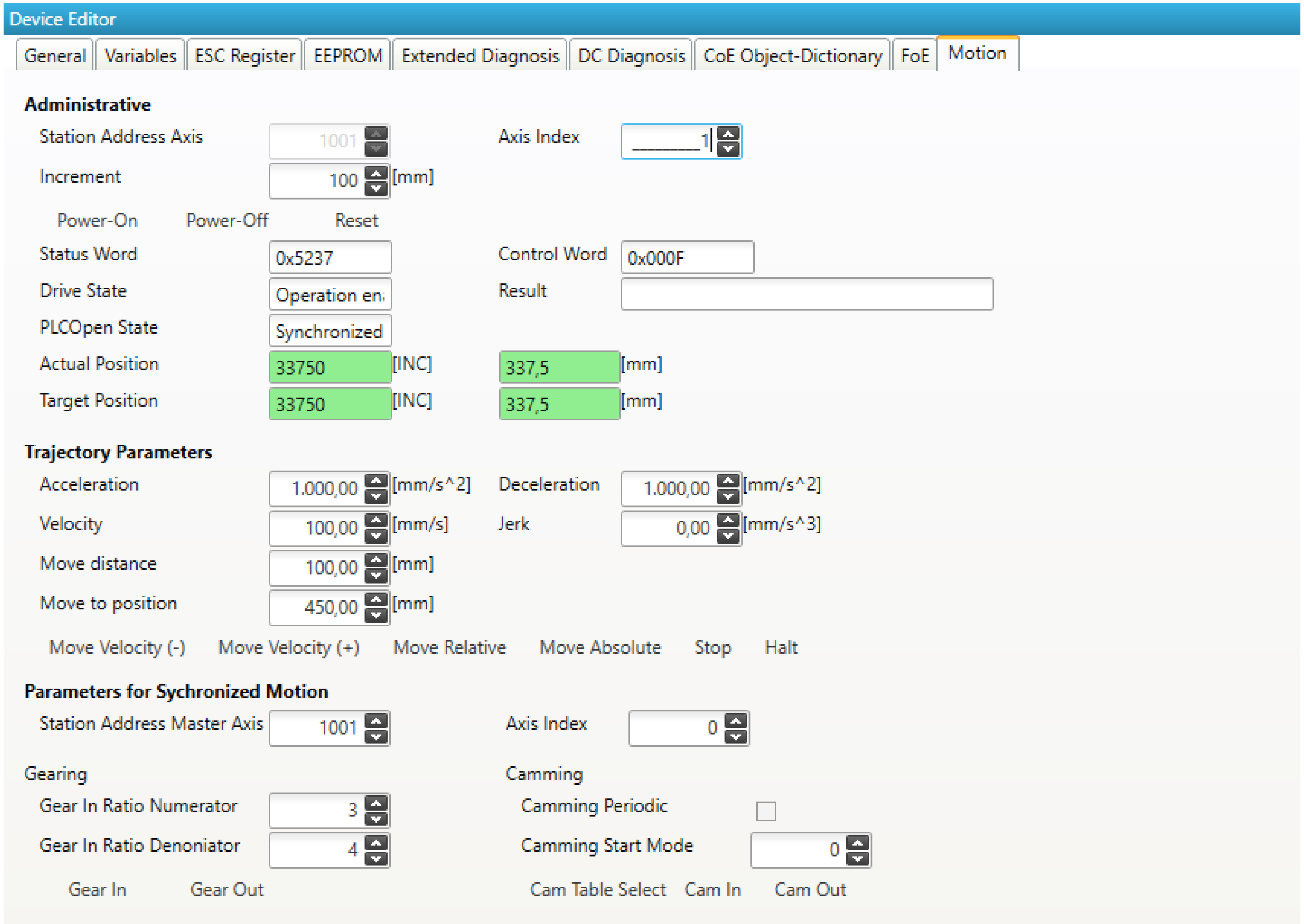
Gear In Ratio Numerator: 3
Gear In Ratio Denoniator: 4

Camming

Camming Periodic ☐
Camming Start Mode: 0



Example EcMasterDemoMotion
Remote Control with EC-Engineer



Administrative

Station Address Axis: 1001
Axis Index: 1
Increment: 100 [mm]

Status Word: 0x5237
Control Word: 0x000F
Drive State: Operation en.
Result:
PLCOpen State: Synchronized
Actual Position: 33750 [INC] 337,5 [mm]
Target Position: 33750 [INC] 337,5 [mm]

Trajectory Parameters

Acceleration: 1.000,00 [mm/s^2]
Deceleration: 1.000,00 [mm/s^2]
Velocity: 100,00 [mm/s]
Jerk: 0,00 [mm/s^3]
Move distance: 100,00 [mm]
Move to position: 450,00 [mm]

Parameters for Sychronized Motion

Station Address Master Axis: 1001
Axis Index: 0

Gearing
Gear In Ratio Numerator: 3
Gear In Ratio Denoniator: 4

Camming
Camming Periodic ☐
Camming Start Mode: 0

Highlights

- CiA402 Profile
- Jerk limited movements
- Changing parameters during movement (continuous update)
- Software limits
- Buffer modes (buffered, blending)
- Operating modes
  - Cyclic Synchronous Position (CSP)
  - Cyclic Synchronous Velocity (CSV)
  - Profile Position (PP)
- Virtual axis
- Efficient implementation → Low CPU load
- Library includes source code