acontis technologies GmbH

SOFTWARE

EC-Motion

PLCopen conformant motion control for EtherCAT

Version 2.1

Edition: 2016-08-09

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

© Copyright **acontis technologies GmbH**

Neither this document nor excerpts therefrom may be reproduced, transmitted, or conveyed to third parties by any means whatever without the express permission of the publisher. At the time of publication, the functions described in this document and those implemented in the corresponding hardware and/or software were carefully verified; nonetheless, for technical reasons, it cannot be guaranteed that no discrepancies exist. This document will be regularly examined so that corrections can be made in subsequent editions. Note: Although a product may include undocumented features, such features are not considered to be part of the product, and their functionality is therefore not subject to any form of support or guarantee.

Table of content

Table of content ... 3
0 Definitions ... 5
0.1 Abbreviations ... 5
1 Product Overview ... 6
1.0 Implemented MCFB's ... 7
2 Example Framework ... 8
2.0 Building Blocks of an EC-Motion control system ... 8
2.1 Running the example program ... 10
2.1.1 Windows (non realtime) ... 10
2.1.2 Linux (RT_PREEMPT) ... 10
2.1.3 VxWorks ... 10
2.2 DemoConfig.xml ... 11
2.2.1 Drive profile parameters ... 11
2.2.2 Motion parameters ... 11
2.2.3 EC-Master related parameters ... 12
2.2.4 Advanced parameters ... 12
3 Programmer's Guide ... 13
3.0 Typical API usage example ... 13
3.1 Profile Position Mode ... 14
3.2 Homing ... 15
3.3 Platform notes ... 15
3.3.1 VxWorks 6.9 on PowerPC e500v2 ... 16
3.3.2 Other platforms ... 16
4 API Reference ... 17
4.0 Initialization ... 17
4.0.1 MC_T_AXIS_REF type ... 17
4.0.2 MC_T_AXIS_INIT type ... 19
4.0.3 MC_T_AXIS_INIT_INPUTS type ... 19
4.0.4 MC_T_AXIS_INIT_OUTPUTS type ... 19
4.0.5 MC_T_AXIS_INIT_ECAT type ... 20
4.1 Common API for MCFB's ... 22
4.1.1 Mapping of MCFB's to C++ classes ... 22
4.2 Single Axis Function Blocks ... 23
4.2.1 Motion MCFB's ... 23
4.2.2 Administrative MCFB's ... 31
4.3 Camming Function Blocks ... 46
4.3.1 CAM Table ... 46
4.3.2 Calculation of Slave axis position ... 47
4.3.3 MC_CAMTABLE_SELECT_T ... 49
4.3.4 MC_CAM_IN_T ... 51
4.3.5 MC_CAM_OUT_T ... 53
4.4 Extension functions ... 54
4.4.1 AMC_CHECK_TARGETPOS_REACHED_T ... 54
4.4.2 AMC_HALT_RECOVERY_T ... 54
4.4.3 MC_CalcMoveProfile ... 55

4.4.4 MC_CalcMoveTimeAtPos 56
4.4.5 MC_DriveSetTargetStep 56
4.4.6 ELMO extension functions 58
Bibliography 61
Appendix A 62

0 Definitions

0.1 Abbreviations

Term	Description
CANopen®	CiA's CAN application layer protocol.
CiA	**C**AN **i**n **A**utomation. can-cia.org
CiA 402	CiA's "*CANopen device profile for drives and motion control*". A.k.a. DS 402.
CoE	**C**ANopen **o**ver **E**therCAT.
CSP	"**C**yclic **S**ynchronous **P**osition mode". Operation mode of CiA 402 drives. => Position control.
CST	"**C**yclic **S**ynchronous **T**orque mode". Operation mode of CiA 402 drives.=> Torque control.
CSV	"**C**yclic **S**ynchronous **V**elocity mode". Operation mode of CiA 402 drives. => Velocity control.
Drive	EtherCAT connected servo drive controller.
ETG	**E**therCAT **T**echnology Group. ethercat.org
IEC	**I**nternational **E**lectrotechnical **C**ommission. iec.ch
MCFB	**M**otion **C**ontrol **F**unction **B**lock. Applies in this document also to the implementation as C++ class. Described in PLCopen group *"Function blocks for motion control, version 2.0"* specification.
PLC (SPS)	**P**rogrammable **L**ogic **C**ontroller / **S**peicher**p**rogrammierbare **S**teuerung.
PLCopen	PLCopen organization. plcopen.org
PP	**"P**rofile **P**osition mode". Operation mode of CiA 402 drives. => Profile position.
SERCOS®	**SE**rial **R**ealtime **CO**mmunication **S**ystem. sercos.de
SoE	**S**ervo **o**ver **E**therCAT. A.k.a. SERCOS over EtherCAT.
STA	acontis' Slave-Test-Application. Remote control and diagnosis tool for EC-Master.

1 Product Overview

EC-Motion is a motion control solution for EtherCAT. The heart of EC-Motion is a C++ class library that implements the PLCopen "*Function blocks for motion control*" specification in version 2.0.
EC-Motion is targeted to work in conjunction with the acontis EC-Master (EtherCAT Master library). But the EC-Master library 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
- SERCOS® / Servo over EtherCAT

Figure 1-1 shows how the different motion control standards play together within EC-Motion. On the EtherCAT Master the MCFB's according to PLCopen are implemented. This is the primary user interface.

The physical communication channel is EtherCAT. As application protocol on top of EtherCAT runs the CiA 402 or SERCOS protocol. These protocols are mapped to EtherCAT according to IEC 61800-7-300 (as "drive profiles").

Under the hood the MCFB's implement the CiA 402 or SERCOS drive profile over EtherCAT. This is mostly a state machine that is processed according to the profile specification.

All internal motion control tasks (trajectory generation, interpolation and state machines) are hidden from the user and abstracted into the MCFB's. E.g. to move an axis for and back you simply instance one MC_MOVE_RELATIVE_T MCFB and set several motion input parameters (Distance to move, maximum velocity, maximum acceleration and maximum jerk). After you set the Execute flag the motion is executed and if the move has finished, the Done output is set.

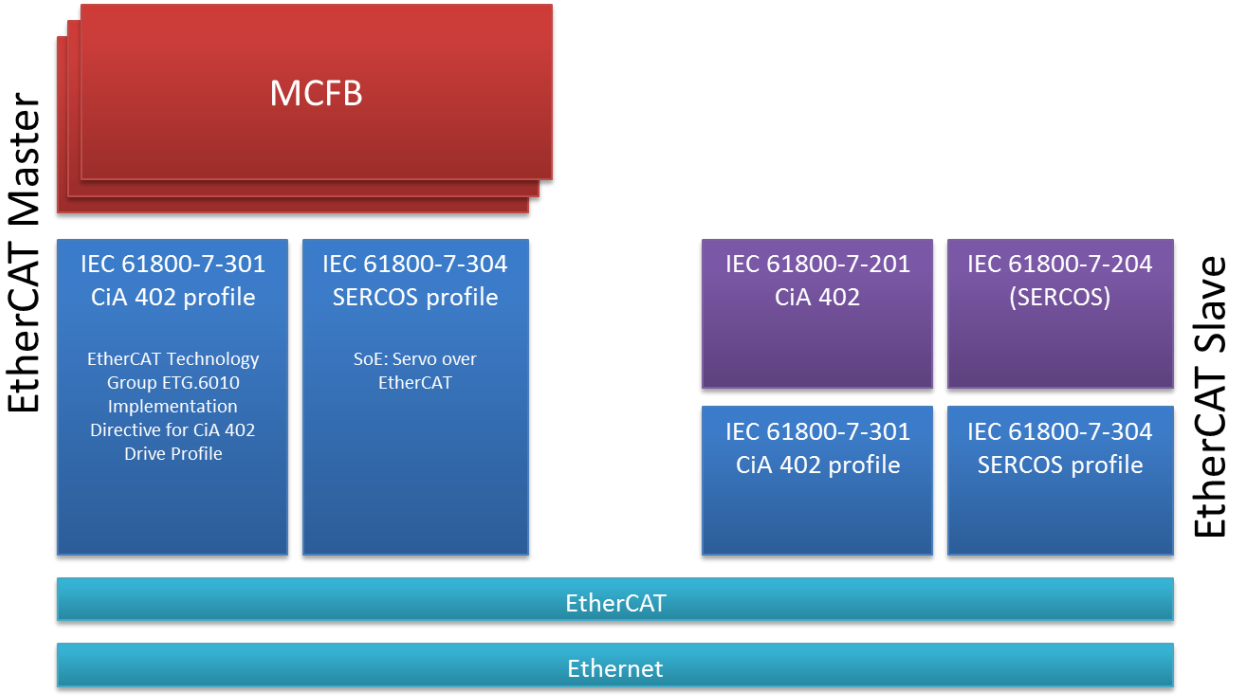


Figure 1-1 EtherCAT servo drive profiles

For optimal use of EC-Motion, it is highly recommended to familiarize yourself with the following documents:

Document	Organization

Function blocks for motion control [1]	PLCopen
EC-Master manual [2]	acontis technologies GmbH
User's manual of your drive	Your EtherCAT servo drive vendor

1.0 Implemented MCFB's

The following table gives an overview of the implemented MCFB's according to the PLCopen standard:

Single Axis Function Blocks	Supported	Comments (<= 48 char.)
MC_POWER_T	**V2.0**	
MC_HOME_T	**V2.0**	
MC_STOP_T	**V2.0**	
MC_HALT_T	**V2.0**	
MC_MOVE_ABSOLUTE_T	**V2.0**	
MC_MOVE_RELATIVE_T	**V2.0**	
MC_MoveAdditive	*N*	
MC_MoveSuperimposed	*N*	
MC_HaltSuperimposed	*N*	
MC_MOVE_VELOCITY_T	**V2.0**	
MC_MOVE_CONT_ABSOLUTE_T	**V2.0**	
MC_MOVE_CONT_RELATIVE_T	**V2.0**	
MC_TorqueControl	*N*	
MC_PositionProfile	*N*	
MC_VelocityProfile	*N*	
MC_AccelerationProfile	*N*	
MC_SET_POSITION_T	**V2.0**	
MC_SetOverride	*N*	
MC_READ_PARAMETER_T & MC_READ_BOOL_PARAMETER_T	**V2.0**	
MC_WRITE_PARAMETER_T & MC_WRITE_BOOL_PARAMETER_T	**V2.0**	
MC_READ_DIGITAL_INPUT_T	**V2.0**	
MC_READ_DIGITAL_OUTPUT_T	**V2.0**	
MC_WRITE_DIGITAL_OUTPUT_T	**V2.0**	
MC_READ_ACTUAL_POSITION_T	**V2.0**	
MC_READ_ACTUAL_VELOCITY_T	**V2.0**	
MC_ReadActualTorque	*N*	
MC_ReadStatus	*N*	
MC_READ_MOTION_STATE_T	**V2.0**	
MC_READ_AXIS_INFO_T	**V2.0**	
MC_READ_AXIS_ERROR_T	**V2.0**	
MC_RESET_T	**V2.0**	
MC_DigitalCamSwitch	*N*	
MC_TouchProbe	*N*	
MC_AbortTrigger	*N*	
MC_CamTableSelect	**V2.0**	
MC_CamIn	**V2.0**	
MC_CamOut	**V2.0**	
MC_GearIn	*N*	
MC_GearOut	*N*	
MC_GearInPos	*N*	
MC_PhasingAbsolute	*N*	
MC_PhasingRelative	*N*	
MC_CombineAxes	*N*	

2 Example Framework

acontis technologies provides the "*EcMasterDemoMotion*" program that serves as an reference implementation of an motion control application (See Figure 2-1). The source code is included into the EC-Motion product (Examples\ECMasterDemoMotion) and can be extended or used as a starting point for an own motion control application.

2.0 Building Blocks of an EC-Motion control system

Figure 2-1 shows a typical integration of the EC-Motion C++-library together with the EC-Master C-library to turn 3 servo drives that are connected to the EtherCAT fieldbus.

Typically for motion control are cycle times between 100us up to 4ms. For getting a deterministic realtime response, a realtime OS is highly recommended. In this example "ECMasterDemoMotion" runs as realtime task on top of the acontis realtime extension / EtherCAT-bundle "EC-Win".

The ECMasterDemoMotion program is statically linked with the EC-Motion C++-library and dynamically linked with the EC-Master C-library. ECMasterDemoMotion also starts the Remote Access Server (RAS Server) that is integrated into the EC-Master library. The RAS Server is listening on a TCP/IP socket for "Motion Commands" that are send from the EC-SlaveTestApplication (abbr. STA, see Figure 2-3) on user request.

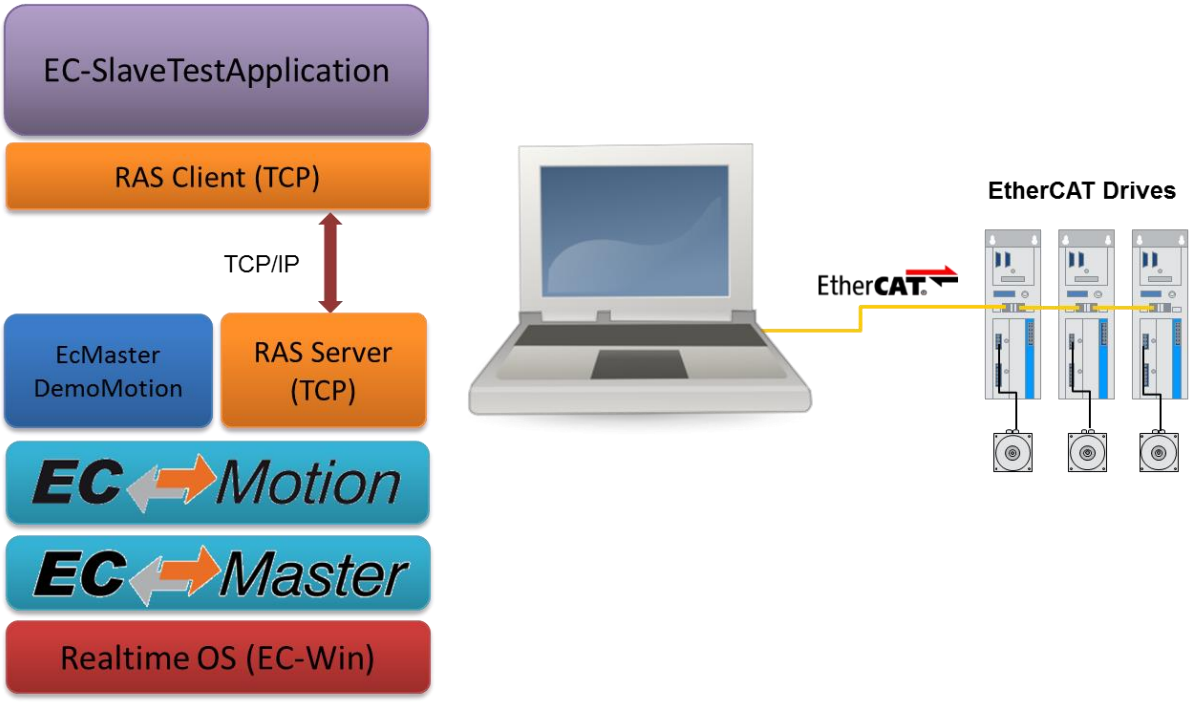


Figure 2-1 Building blocks of an EC-Motion control system

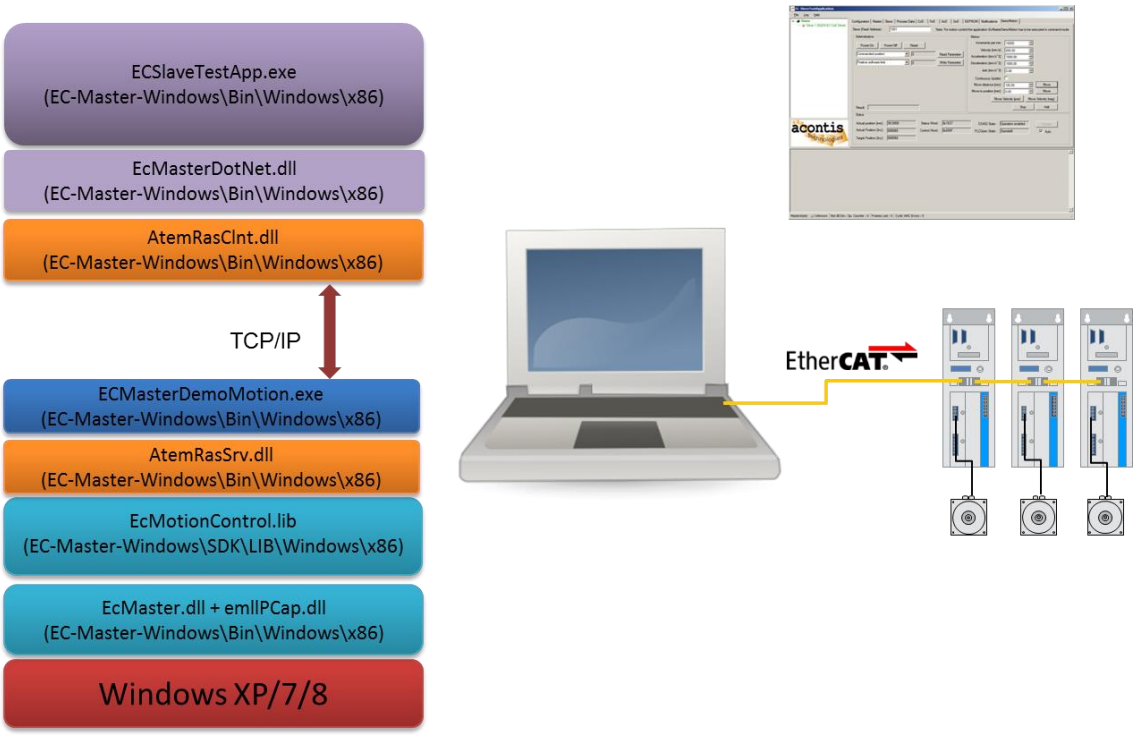


Figure 2-2 EC-Motion software components

The STA is shown in Figure 1-1. Please refer to the EC-Motion quick start guide [3] or the EC-Master manual [2] for usage instructions.

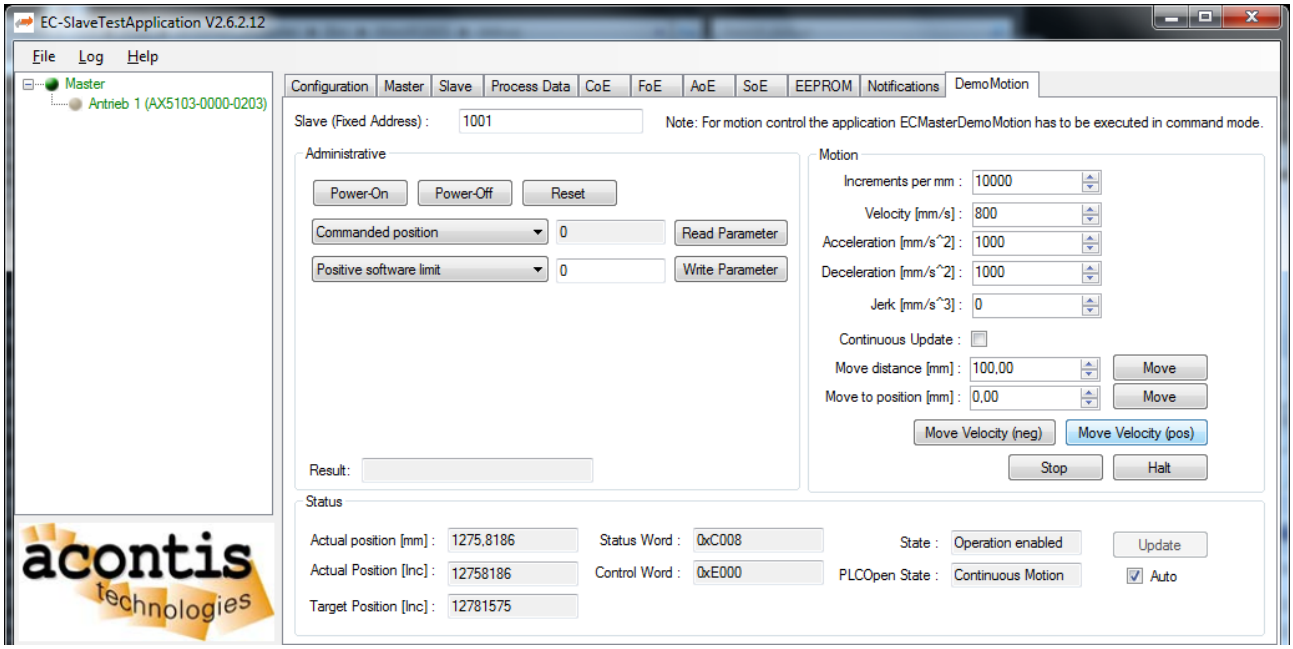


Figure 2-3 STA with motion control pane

Figure 2-4 shows the different files read and written by the ECMasterDemoMotion program.

The program is parametrized by reading a configuration file (see ch. 2.2 for an description of the configuration parameters). Additionally the **E**therCAT **N**etwork **I**nformation (ENI) is read. The name of the ENI comes out of the Cfg.xml. The ENI need to be generated in advance by an configuration tool (e.g. acontis' EC-Engineer) and describes the EtherCAT network configuration.

As outputs we get the log file with status and error printouts.
Additionally there is the „DCM logfile", which is in Comma-Seperated-Value (CSV) format. It records the quality of the „Distributed Clock Master Synchronization" (DCM) controller of the EC-Master. If your timebase is bad, the controller may swing and this file can be used for diagnosis purposes.

Third, the trajectory can be printed out to the „Motion Logfile" (CSV format as well). Importing this file into any spreadsheet program, allows you to visualize the trajectory (Path, Velocity, Acceleration, …).

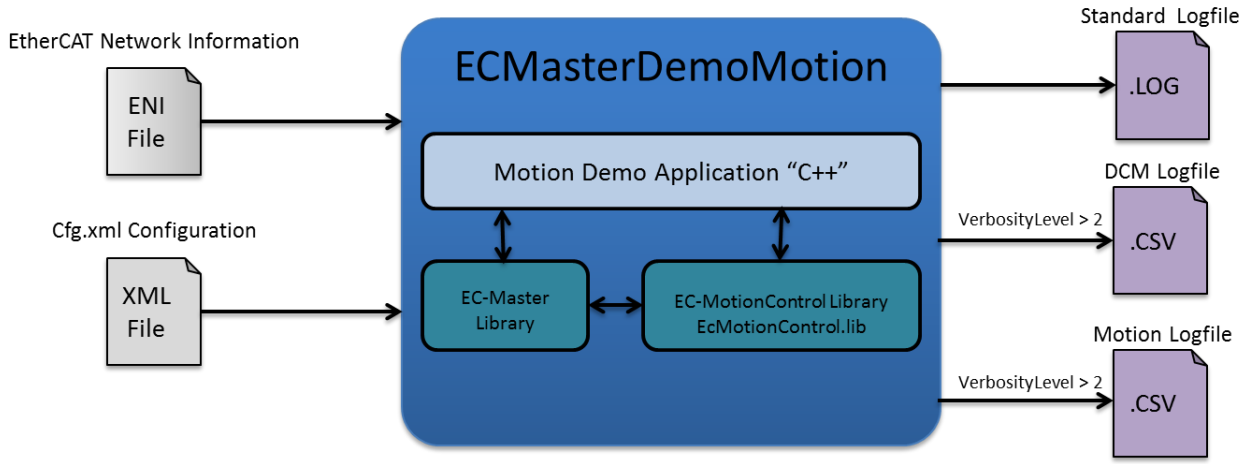


Figure 2-4 Input- and output-files of the ECMasterDemoMotion program

2.1 Running the example program

The general syntax for starting the program is:

EcMasterDemoMotion DemoConfig.xml

The XML file DemoConfig.xml holds all the parameters the example program need to know about. It should be modified to fit your environment.
In the EC-Motion package are already sample configuration files included (Examples\ECMasterDemoMotion\Config) that can be modified to your needs.

Below are call syntaxes for selected platforms. For platforms not listed here, please refer to the EC-Master manual and look up how the regular EC-MasterDemo is started on that particular platform.

2.1.1 Windows (non realtime)

EcMasterDemoMotion DemoConfigEval.xml

2.1.2 Linux (RT_PREEMPT)

./EcMasterDemoMotion DemoConfigEval.xml

2.1.3 VxWorks

ld < emllETSEC.out

ld < MotionDemo.out
sp motionDemo, "DemoConfigEval.xml"

2.2 DemoConfig.xml

EcMasterDemoMotion supports two operating modes:

- Command mode: Motion is remotely commanded with the SlaveTestApplication (see Figure 2-3). => Set parameter "*Config/MotionDemo/CmdMode*" to 1.

- Standalone mode: The example program runs standalone and turns the configured axes forward and backward (uses MCFB MoveRelative). => Set parameter "*Config/MotionDemo/CmdMode*" to 0.

EcMasterDemoMotion supports up to 4 axes without recompilation, but you can change the `DEMO_MAX_NUM_OF_AXIS` define to support more than 4 axes.
In the command mode only the first axis per drive can be commanded remotely.

2.2.1 Drive profile parameters

You should adapt the "Config/MotionDemo/DriveN/IdxXXX" parameters to the correct variable indices. The format is "0xXXX:0xYYY" where XXX is the index of the PDO in hexadecimal and YYY the index of the PDO-variable within the PDO in hexadecimal. If the format is "0xYYY", the content is interpreted as the index to the first PDO-variable with this index.

Additionally the parameter "*Config/MotionDemo/DriveN/OperationMode*" can be changed if you prefer a different operating mode.

> You should take care to include the above mentioned PDO's and PDO-variables inside the EtherCAT configuration tool (e.g. EC-Engineer) and export the ENI file.
>
> Please read the manual of your servo drive controller for detailed informations about the different drive operating modes.

2.2.1.1 CiA 402 drive

DemoConfigEval.xml can be used as template.
Set parameter "*Config/MotionDemo/DriveN/DriveProfile*" to "DS402" and adapt "*Config/MotionDemo/DriveN/CoeIdxOpMode*" if needed.

2.2.1.2 SERCOS / SoE drive

DemoConfig_ax5103.xml can be used as template.
Set parameter "*Config/MotionDemo/DriveN/DriveProfile*" to SERCOS and "*Config/MotionDemo/DriveN/SercosDriveNo*" to the drive number inside your EtherCAT slave (normally 0 for the first drive, 1 for the second).

2.2.2 Motion parameters

The following parameters are only evaluated in the standalone mode. If the command mode is active, these values are set up and transmitted from the STA.

- The parameters "*Config/MotionDemo/DriveN/Vel*", "*Acc*", "*Dec*" and "*Jerk*" control the maximum values for velocity, acceleration, decceleration and jerk in [u]/s.

- The parameter "*Config/MotionDemo/DriveN/IncPerMM*" controls the increments per physical unit [u]. It influences the Vel, Acc, Dec, Jerk and Distance input values.

E.g.: Your drive actual position is incremented with 80000 per physical revolution. Setting "*Config/MotionDemo/DriveN/IncPerMM*" to 80000 will turn your drive with 120 RPM (2 revolutions per second) if you set the velocity to 2 [u]/s => [u] == 80000 => (2 * 80000/s).

- The parameter "*Config/MotionDemo/DriveN/IncFactor*" determines the internal resolution of the motion kernel. The kernel does floating point calculation only if the trajectory is updated. For running the interpolator in each cycle, fixpoint mathematics is used. For avoiding quantization errors, the internal values are scaled with "*Config/MotionDemo/DriveN/IncFactor*". So if you see poor resolution (e.g. drive will not move at very low velocities), increment this value one by one and retest.

- If the drive is operated in velocity operation mode (CSV) you may change the parameter "*Config/MotionDemo/DriveN/VelocityGain*" to increase the gain of the P-controller. This may be needed if the drive lags.

2.2.3 EC-Master related parameters

Normally the following parameters need to be adapted for your environment:

- "*Config/MotionDemo/DriveN/Address*". EtherCAT station address.

- "*Config/Common/BusCycleTime*". Cycle period of the cyclic task in microseconds. Default is 1000.

- "*Config/Common/AuxClk*". Cycle period of the cyclic task in microseconds. Default is 1000. Set to 0 if the auxillary clock (Hardware interrupt timer) is not supported for this particular platform.

- "*Config/Common/LinkLayer*". Initialization string for the MAC driver. Please see the EC-Master manual [2], chap. 3.6 for details.

- "*Config/Common/ENIFileName*". Path to the EtherCAT Network Information (ENI) file. See chap. 2.1.2. of the EC-Master manual [2].

- "*Config/Common/VerbosityLevel*". Verbosity level for log messages. Default is 2.

2.2.4 Advanced parameters

Normally there is no need to change the following parameters:

- "*Config/Common/RASEnabled*". If 0, the RAS server is not started and remote control is not possible. Default is 1 (enabled).

- "*Config/Common/RASPort*". The TCP/IP socket port number for the RAS server. Default is 6000.

- "*Config/Common/CpuAffinity*". Index of the CPU on which the various threads are running. Default is 0 (first CPU). 1 is CPU2, 2 is CPU3, …

- "*Config/Common/DemoDuration*". How long in seconds the program should run.

- "*Config/Common/PerfMeasurement*". Enable tracing of performance related data. Default is 1.

- "*Config/MotionDemo/NoDCMBusShift*". Disable Distributed Clocks Master (DCM) bus shift controller. Default is 1. See the EC-Master manual [2] for details. Default is 0.

- "*Config/MotionDemo/DCMCtlSetVal*". DCM controller set value in nanoseconds. Default is 2500000. See the EC-Master manual [2] for details. Default is 0.

3 Programmer's Guide

3.0 Typical API usage example

The following example shows the basic API usage flow. This example is not complete! Please refer to chapter API Reference for a detailed description of the API.

Link to the static EC-Motion library (`EcMotionControl.lib / libEcMotionControl.a`) and:

```
#include <EcMotionControl.h>
```

Do basic initialization of MC_T_AXIS_REF. E.g.:

```
MC_T_AXIS_INIT AxisInitData = { 0 };
AxisInitData.CycleTime = 1000;
AxisInitData.IncPerMM = 80000;
// ... do more initialization
MC_T_AXIS_REF Axis;
Axis.Init(AxisInitData);
```

Initialize inputs and outputs:

```
MC_T_AXIS_INIT_INPUTS inp = { 0 };
OsMemset(&inp, 0, sizeof(MC_T_AXIS_INIT_INPUTS));
inp.pActualPosition = plPdActualPosition; // Points into process data memory
Axis.InitInputs(inp);

MC_T_AXIS_INIT_OUTPUTS outp = { 0 };
OsMemset(&outp, 0, sizeof(MC_T_AXIS_INIT_OUTPUTS));
outp.pTargetPosition = plPdTargetPosition; // Points into process data memory
Axis.InitOutputs(outp);
```

Do EtherCAT specific initialization:

```
MC_T_AXIS_INIT_ECAT EcatInit = { 0 };
OsMemset(&EcatInit, 0, sizeof(MC_T_AXIS_INIT_ ECAT));
EcatInit.dwProductCode = 2;
// ... do more initialization
EcatInit.eProfile = MC_T_AXIS_PROFILE_DS402;
Axis.InitEcat(EcatInit);
```

Set mode of operation:

```
// Precondition: EtherCAT slave must be in preoperational state
Axis.SetModeOfOperation(8); // CiA 402 position mode (CSP)
```

Create MCFB's. Note: MCFB's are implemented as C++ classes. Don't memset to zero:

```
MC_POWER_T                  Power; // Mandatory!
MC_HALT_T                   Halt;
MC_MOVE_RELATIVE_T          MoveRelative;
// ... create additional MCFB's
```

Initialize MCFB's INPUT data:

```
Power.Axis = &Axis;
Halt.Axis = &Axis;
MoveRelative.Axis = &Axis;
MoveRelative.Distance = 10.0; // Turn motor forward
MoveRelative.Velocity = 100.0;
MoveRelative.Acceleration = 500.0;
```

```
MoveRelative.Jerk = 0.0;
// ... do more initialization
Power.Enable = Power.EnablePositive = Power.EnableNegative = MC_TRUE;
```

Run cyclic operation. E.g.:

```
for (;;)
{
   // Wait for next cycle

   Power.OnCycle(); // Mandatory
   Halt.OnCycle();
   MoveRelative.OnCycle();
   // ... process additional MCFB's

   if (Power.Status) MoveRelative.Execute = MC_TRUE;

   if (MoveRelative.Done)
   {
      MoveRelative.Execute = MC_FALSE;
      MoveRelative.OnCycle();
      MoveRelative.Distance = -10.0; // Turn motor backward
      MoveRelative.Execute = MC_TRUE;
   }
}
```

3.1 Profile Position Mode

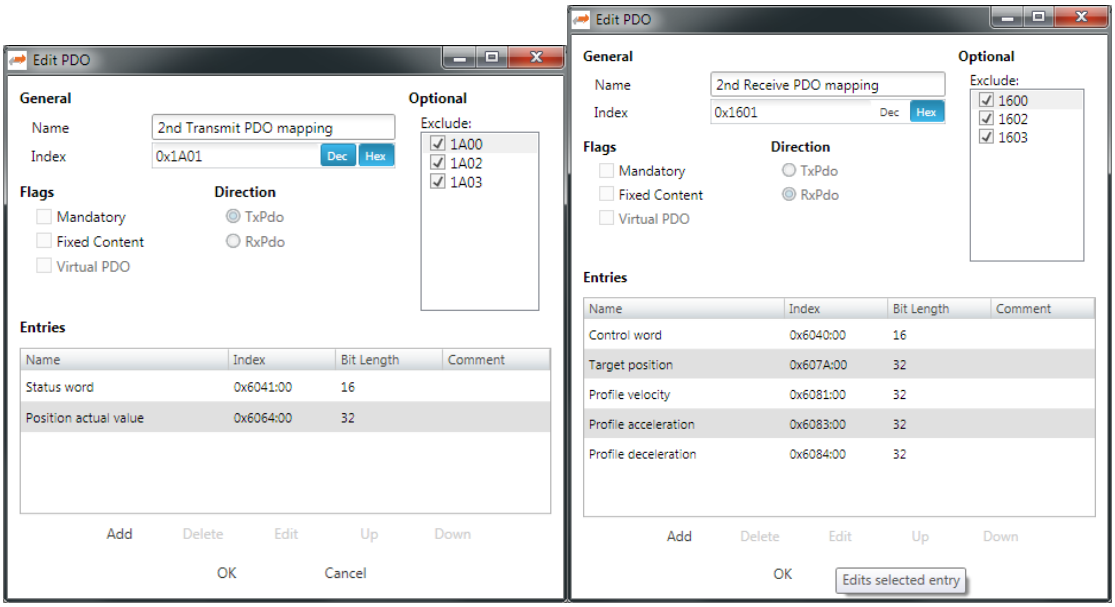
The PP mode has to be activated after general initialization (s. Typical API usage example) using `SetModeOfOperation()`

```
Axis.SetModeOfOperation(DRV_MODE_OP_PROF_POS);
```

Afterwards profile position mode has to be configured like

```
Axis.SetMotionProfileType(PROFILE_TYPE);
Axis.SetProfileOptionCode(PROFILE_OPTION_CODE);
Axis.SetPositionWindow(…, …);
Axis.SetSoftwareLimits(MC_TRUE, MC_TRUE, 1000.0, -450.0);
```

Movements itself will be handled by general FBs like MC_MOVE_ABSOLUTE_T and MC_MOVE_RELATIVE_T. For proper functioning of movement FBs the following EtherCAT objects have to be mapped into PDO:



The axis error code (CiA402 0x603F) can be checked each time with `Axis.GetErrorCode()` (s. MC_T_WORD GetErrorCode). In order to use this function the object 0x603F has to be mapped into a PDO. Due to design only this function can be called within tEcJobTask, the other PP functions cannot be called in tEcJobTask.

```
MC_T_AXIS_INIT_INPUTS inp = { 0 };
OsMemset(&inp, 0, sizeof(MC_T_AXIS_INIT_INPUTS));
inp.pActualPosition = plPdActualPosition;    // Points into process data memory
inp.pDigitalInputs = pdwDigitalOutputs;      // Points to digital inputs in process
data
inp.pErrorCode = pwErrorCode;                // Points to error code (object 0x603f)

Axis.InitInputs(inp);

MC_T_AXIS_INIT_OUTPUTS outp = { 0 };
OsMemset(&outp, 0, sizeof(MC_T_AXIS_INIT_OUTPUTS));
outp.pTargetPosition = plPdTargetPosition;   // Points into process data memory
outp.pProfileVelocity = plPdProfileVelocity; // Points to profile velocity
outp.pProfileAcc = plProfileAcc;             // Points to profile acceleration
outp.pProfileDec = plProfileDec;             // Points to profile deceleration
outp.pModeOfOperation = pbyPdModeOfOperation;// Points to mode of operation (DS402
0x6060
outp.pDigitalOutputs = pdwPdDigitalOutputs;  // Points to digital outputs in process
data

Axis.InitOutputs(outp);
```

3.2 Homing

The homing sequence is performed by FB MC_Home (s. MC_HOME_T). The details of homing sequence are manufacture dependent and can be set by the axis' parameters. The homing axis's parameter will be set by `Axis.SetHomingParameters()`, please note this function may not be called within cyclic task (tEcJobTask).

3.3 Platform notes

The EC-Motion C++ library is written in portable ANSI C++. It uses the platform-abstraction-layer from the EC-Master library. All platforms that are support by EC-Master are supported for EC-Motion as well. Please see the user's manual of EC-Master for a list of currently supported platforms.

3.3.1 VxWorks 6.9 on PowerPC e500v2

For this platform the hardware floating point support must be explicitly enabled.

Replace the following compiler flags:
```
-te500v2 -fno-implicit-fp -mspe=no
```
With:
```
-te500v2 -mcpu=8548 -mfloat-gprs=double -mspe=yes -mabi=spe
```

3.3.2 Other platforms

For optimal performance it is recommended to enable the hardware Floating Point Unit (FPU) support if available.

4 API Reference

Function prototypes, definitions etc. of the API can be found in the header file `EcMotionControl.h` which is the header file to include when using the EC-Motion library.

4.0 Initialization

4.0.1 MC_T_AXIS_REF type

The MC_T_AXIS_REF C++ class matches with the AXIS_REF type in the PLCopen specification. Each controlled axis needs an own instance of MC_T_AXIS_REF.

This class should be considered as an opaque handle. Don't make any assumptions about the data layout. It must be initialized before passed to the MCFB's with the following initialization methods:

4.0.1.1 Init(const MC_T_AXIS_INIT &)

Do basic initialization. This must be called at least once prior calling the first OnCycle() for any MCFB.

4.0.1.2 InitInputs(const MC_T_AXIS_INIT_INPUTS &)

Initialize the pointers to the input data. For EtherCAT operation this must be called at least once prior calling the first `OnCycle()` for any MCFB.

Usually pointers into the process data memory of the EtherCAT-Master are passed that point to the drive status word and the drive's actual position word.

If `MC_T_AXIS_INIT::AxisType` is set to `MC_AXIS_TYPE_VIRTUAL` (simulated axis) then this call is optional. If called, the actual position is read from the memory location the pointer points to.

4.0.1.3 InitOutputs(const MC_T_AXIS_INIT_OUTPUTS &)

Initialize the pointers to the output data. For EtherCAT operation this must be called at least once prior calling the first `OnCycle()` for any MCFB.

Usually pointers into the process data memory of the EtherCAT Master are passed that point to the drive's control word and the drive desired position word.

If `MC_T_AXIS_INIT::AxisType` is set to `MC_AXIS_TYPE_VIRTUAL` (simulated axis) then this call is optional. If called, the target position is written to the memory location the pointer points to.

4.0.1.4 InitEcat(const MC_T_AXIS_INIT_ECAT &)

Initialize axis for EtherCAT operation. For EtherCAT operation this must be called at least once prior calling the first `OnCycle()` for any MCFB. For non EtherCAT operation, this call shall be omitted.

4.0.1.5 EC_T_DWORD SetModeOfOperation(EC_T_WORD)

Sets drive operation mode. If `MC_T_AXIS_INIT::AxisType` is set to `MC_AXIS_TYPE_VIRTUAL` the parameter shall be `OPMODE_CSP` or `OPMODE_CSV`.
If `MC_T_AXIS_INIT::AxisType` is `MC_AXIS_TYPE_REAL_ALL` the parameter is any of `MC_T_CIA402_OPMODE` (CiA 402 drive profile) or `MC_T_SERCOS_OPMODE` (SERCOS drive profile).

This call is mandatory and shall be called after the InitXXX() function have been called.

For non-simulated axis: This method should only be called if the EtherCAT slave is in pre operational state, otherwise != 0 is returned (Error).
This function returns an error code in case of error otherwise MC_NO_ERROR.

4.0.1.6 MC_T_DWORD SetPositionWindow(MC_T_DWORD dwWindow, MC_T_WORD wTime)

Sets position window (CiA402 0x6067) to `dwWindow` and position window time (CiA402 0x6068) to `wTime` for position functions. This function returns an error code in case of error otherwise MC_NO_ERROR.

4.0.1.7 MC_T_DWORD SetHomingParameters(MC_T_BYTE byMethod, MC_T_REAL fSpeedSearchSwitch, MC_T_REAL fSpeedSearchZero, MC_T_REAL fAcceleration, MC_T_REAL fOffset)

Sets axis parameter for homing sequence:
- byMethod, homing method, CiA402 0x6098
- fSpeedSearchSwitch, speed during search for switch [mm/s]
- fSpeedSearchZero, speed during search for zero [mm/s]
- fAcceleration, homing acceleration [mm/s²]
- fOffset, home offset [mm]

The homing parameter must be set before calling MC_Home().

4.0.1.8 MC_T_DWORD GetErrorCode(MC_T_WORD* pwErrorCode)

Returns code of the last error occurred in device (CiA402 0x603F). The error code object has to be mapped to into a PDO. InitInputs -> pointer setzen.

4.0.1.9 MC_T_DWORD SetSoftwareLimits(MC_T_BOOL bEnaLimitPos, MC_T_BOOL bEnaLimitNeg, MC_T_REAL fLimitPos, MC_T_REAL fLimitNeg);

Sets axis software limits:
- bEnaLimitPos, MC_TRUE = the positive axis limit will be set from fLimitPos, MC_FALSE = fLimitPos will be ignored and the positive axis limit will be set to 0
- bEnaLimitNeg, MC_TRUE = the negative axis limit will be set from fLimitPos, MC_FALSE = fLimitPos will be ignored and the negative axis limit will be set to 0
- fLimitPos, positive axis limit [mm]
- fLimitNeg, negative axis limit [mm]

4.0.1.10 MC_T_DWORD SetProfileOptionCode(MC_T_WORD wPosOptionCode)

Sets positioning option code (CiA402 0x60F2) from `wPosOptionCode`.
This function returns an error code in case of error otherwise MC_NO_ERROR.

4.0.1.11 MC_T_DWORD SetMotionProfileType (MC_T_SWORD swProfileOperationMode)

Sets motion profile type (CiA402 0x6086) from `swProfileOperationMode`.
This function returns an error code in case of error otherwise MC_NO_ERROR.

4.0.2 MC_T_AXIS_INIT type

C structure. Can be memset to zero before first usage.

Type	Variable	Description
MC_T_AXIS_TYPE	AxisType	MC_AXIS_TYPE_VIRTUAL (Simulated axis) or MC_AXIS_TYPE_REAL_ALL (EtherCAT operation)
MC_T_DWORD	CycleTime	PLC / EtherCAT cycle time in microsecond.
MC_T_DWORD	IncPerMM	Conversion factor for the output of the position / velocity values. Usual these are increments per mm or axis revolution. E.g. 80000 if the drive's encoder (drive actual value) counts 80000 increments per revolution.
MC_T_DWORD	IncFactor	Scaling factor for the internal position calculation. The internal position values are shifted left with this factor (PosRaw << IncFactor). This factor should be increased to improve the internal resolution (e.g. if the axis will not turn with low velocities).
MC_T_DWORD	Verbose	Verbosity for logging messages. 1, 2, …
MC_T_DWORD	PosWindow	In position window in increments.
MC_T_DWORD	VelocityGain	Velocity gain factor for the velocity operation mode (CSV).

4.0.3 MC_T_AXIS_INIT_INPUTS type

C structure. Can be memset to zero before first usage.

Type	Variable	Description
MC_T_SDWORD	pActualPosition	Pointer to a signed 32 Bit variable that holds the drive's actual position (drive encoder/resolver). The content of the variable is read once per cycle.
MC_T_SWORD	pActualTorque	Pointer to a signed 16 Bit variable (–32768 to +32767) that holds the actual torque in process data, MC_NULL if torque mode is not used.
MC_T_DWORD	pDigitalInputs	Pointer to digital inputs in process data

4.0.4 MC_T_AXIS_INIT_OUTPUTS type

C structure. Can be memset to zero before first usage.

Type	Variable	Description
MC_T_SDWORD	pTargetPosition	Pointer to a signed 32 Bit variable that holds the drive's set position. The content of the variable is updated once per cycle if the API operates in position mode (CSP). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_SDWORD	pTargetVelocity	Pointer to a signed 32 Bit variable that holds the drive's set velocity. The content of the variable is updated once per cycle if the API operates in velocity mode (CSV). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_SDWORD	pVelocityOffset	Pointer to a signed 32 Bit variable that holds the velocity offset for feed forward. The content of the variable is updated once per cycle if the API operates in velocity mode (CSV). See also MC_T_AXIS_REF::SetModeOfOperation().

MC_T_SWORD	pTargetTorque	Pointer to a signed 16 Bit variable that holds the drive's torque. The content of the variable is updated once per cycle if the API operates in torque mode (CST). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_SWORD	pTorqueOffset	Pointer to a signed 16 Bit variable that holds the torque offset for feed forward. The content of the variable is updated once per cycle if the API operates in velocity mode (CSV). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_DWORD	pProfileVelocity	Pointer to an unsigned 32 Bit variable that holds the profile velocity. The content of the variable is updated once per cycle if the API operates in profile position mode (PP). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_DWORD	pProfileAcc	Pointer to an unsigned 32 Bit variable that holds the profile acceleration. The content of the variable is updated once per cycle if the API operates in profile position mode (PP). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_DWORD	pProfileDec	Pointer to an unsigned 32 Bit variable that holds the profile deceleration. The content of the variable is updated once per cycle if the API operates in profile position mode (PP). See also MC_T_AXIS_REF::SetModeOfOperation().
MC_T_BYTE	pModeOfOperation	Pointer to an unsigned 8 Bit variable that holds the mode of operation (DS402 0x6060)
MC_T_DWORD	pDigitalOutputs	Pointer to digital outputs in process data. Can be used in any mode of operation.

4.0.5 MC_T_AXIS_INIT_ECAT type

C structure. Can be memset to zero before first usage.

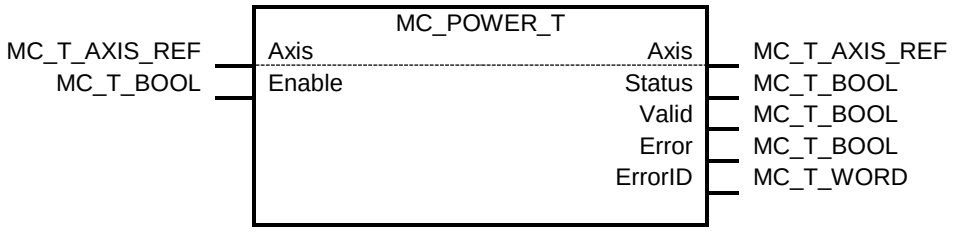
Type	Variable	Description
MC_T_DWORD	VendorId	Vendor ID of the EtherCAT-slave. This info can be obtained with the EC-Master API emGetSlaveInfo().
MC_T_DWORD	ProductCode	Product code of the EtherCAT-slave. This info can be obtained with the EC-Master API emGetSlaveInfo().
MC_T_DWORD	SlaveID	Internal EC-Master EtherCAT-slave identifier. This info can be obtained with the EC-Master API emGetSlaveInfo().
MC_T_WORD	StationAddress	Station address / physical address of the EtherCAT-slave. This info can be obtained with the EC-Master API emGetSlaveInfo().
MC_T_AXIS_PROFILE	Profile	If MC_T_AXIS_INIT::AxisType is MC_AXIS_TYPE_REAL_ALL: MC_T_AXIS_PROFILE_DS402 for CAN over EtherCAT (CoE) drives that support the CiA 402 profile or MC_T_AXIS_PROFILE_SERCOS for SERCOS over EtherCAT / Servo over EtherCAT (SoE) drives. Shall be set to MC_T_AXIS_PROFILE_NONE if the axis is in simulation mode (MC_T_AXIS_INIT::AxisType is MC_AXIS_TYPE_VIRTUAL.
MC_T_BYTE	SercosDriveNo	If Profile is MC_T_AXIS_PROFILE_SERCOS, the addressed SERCOS drive number (0..n) within the

		SERCOS servo controller or 0 otherwise.
MC_T_WORD	CoeIdxOpMode	If Profile is MC_T_AXIS_PROFILE_DS402, the index of the CoE objects for settings the drive's operation mode, 0 otherwise. For the first CoE axis this is usually 0x6060.
MC_T_WORD *	pStatusWord	Pointer to an unsigned 16 Bit variable that holds the drive status word. The content of the variable is read once per cycle for processing the drive's state machine.
MC_T_WORD *	pControlWord	Pointer to an unsigned 16 Bit variable that holds the drive control word. The content of the variable is written for controlling the drive's state machine.
EcatCoeSdoDownloadFptr	pEcatCoeSdoDownload	Pointer to the ecatCoeSdoDownload API function of EC-Master. Mandatory if MC_AXIS_TYPE_REAL_ALL is MC_T_AXIS_PROFILE_DS402, otherwise it shall be set to MC_NULL.
EcatCoeSdoUploadFptr	pEcatCoeSdoUpload	Pointer to the ecatCoeSdoUpload API function of EC-Master. Mandatory if MC_AXIS_TYPE_REAL_ALL is MC_T_AXIS_PROFILE_DS402, otherwise it shall be set to MC_NULL.
EcatSoeWriteFptr	pEcatSoeWrite	Pointer to the ecatSoeWrite API function of EC-Master. Mandatory if MC_AXIS_TYPE_REAL_ALL is MC_T_AXIS_PROFILE_SERCOS, otherwise it shall be set to MC_NULL.
EcatSoeReadFptr	pEcatSoeRead	Pointer to the ecatSoeRead API function of EC-Master. Mandatory if MC_AXIS_TYPE_REAL_ALL is MC_T_AXIS_PROFILE_SERCOS, otherwise it shall be set to MC_NULL.
EcatGetSlaveStateFptr	pEcatGetSlaveState	Pointer to the ecatGetSlaveState API function of EC-Master.

4.1 Common API for MCFB's

4.1.1 Mapping of MCFB's to C++ classes

In chapter Single Axis Function Blocks the following style is used to describe a MCFB:



On the left side are the **input** variables, on the right side the **output** variables. Inputs are read/write. Outputs are read only. The "Axis" variable is **input/output** and should be set once after the MCFB class is instantiated.

The above MCFB maps to the following simplified C++ code:

```
class MC_POWER_T : public MC_FB_T
{
public:

    const MC_T_BOOL     &Status;          // OUT(B): Effective state of the power stage
    const MC_T_BOOL     &Valid;           // OUT(E): If TRUE a ...

    MC_T_BOOL           Enable;           // IN(B): As long as is true, power is on

    MC_POWER_T()
        : MC_FB_T(MCFB_ID_POWER),
          // Init external OUT's
          Status(static_cast<const MC_T_BOOL &>(bStatus)),
          Valid(static_cast<const MC_T_BOOL &>(bValid)),
          // Init variables
          Enable(0),
          bStatus(0),
          bValid(0),
          // Init internal IN's
          bEnable(static_cast<const MC_T_BOOL &>(Enable)) {}

    void OnCycle();

private: // Private state (not shown here)

};
```

All MCFB classes are directly or indirectly inherited from `MC_FB_T` or `MC_BUFFERED_FB_T` which provides variables that are common to several MCFB's. Each MCFB has the following method:

```
void MC_FB_T::OnCycle();
```

This method runs the cyclic part of the MCFB and must be called in each (PLC) cycle at least once.

> The MC_POWER_T MCFB computes the motion trajectory and is mandatory. There must be one and only one instance per axis.
>
> The OnCycle() method has to be called in each cycle.

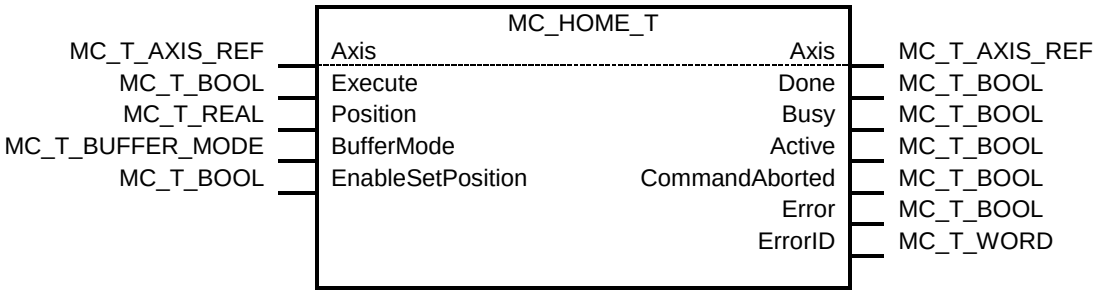
4.2 Single Axis Function Blocks

The following MCFB's are implemented in the EC-Motion C++-library.

4.2.1 Motion MCFB's

4.2.1.1 MC_HOME_T

class		**MC_HOME_T**	
This Function Block commands the axis to perform the "search home" sequence. The details of this sequence are manufacturer dependent and can be set by the axis' parameters. The 'Position' input is used to set the absolute position when reference signal is detected. This Function Block completes at 'Standstill' if it was started in 'Standstill'.			
VAR_IN_OUT			
B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	MC_T_BOOL	Start the motion at rising edge
B	Position	MC_T_REAL	Absolute position when the reference signal is detected [u]
E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
E	EnableSetPosition	MC_T_BOOL	If TRUE the parameter 'Position' will be used to set the new homing position after homing sequence done, otherwise the 'Position' will be ignored.
VAR_OUTPUT			
B	Done	MC_T_BOOL	Reference known and set successfully
E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	MC_T_WORD	Error identification
Notes: MC_Home is a generic FB which does a system specified homing procedure which can be constructed by the StepHoming FBs.			



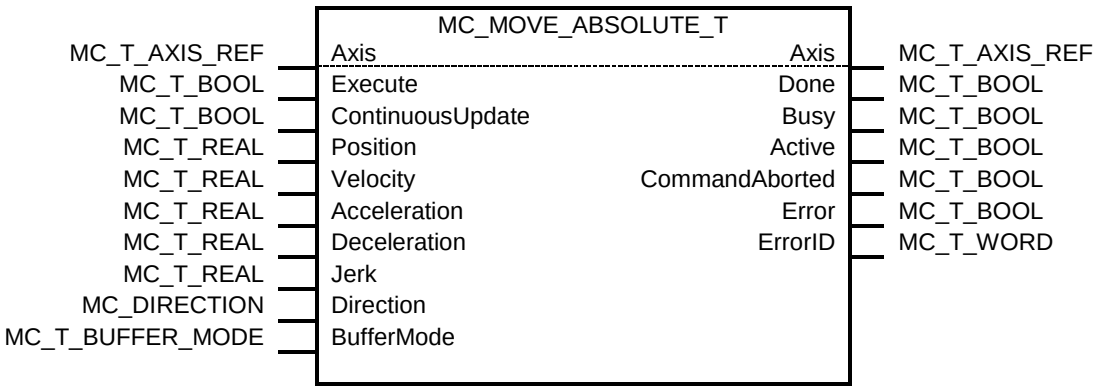
BufferMode is only supported in modes „cyclic synchronous position" and "cyclic synchronous velocity". Homing is supported only in PP mode.

4.2.1.2 MC_MOVE_ABSOLUTE_T

class		**MC_MOVE_ABSOLUTE_T**		
This Function Block commands a controlled motion to a specified absolute position.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start the motion at rising edge
	E	ContinuousUpdate	MC_T_BOOL	Trajectory is continuously updated.
	B	Position	MC_T_REAL	Commanded 'Position' for the motion (in technical unit [u]) (negative or positive)
	B	Velocity	MC_T_REAL	Value of the maximum 'Velocity' (not necessarily reached) [u/s].
	E	Acceleration	MC_T_REAL	Value of the 'Acceleration' (always positive) (increasing energy of the motor) [u/s^2]
	E	Deceleration	MC_T_REAL	Value of the 'Deceleration' (always positive) (decreasing energy of the motor) [u/s^2]
	E	Jerk	MC_T_REAL	Value of the 'Jerk' [u/s^3]. (always positive)
	B	Direction	MC_T_DIRECTION	Enum type (1-of-4 values: `MC_DIR_POSITIVE`, `MC_DIR_SHORTEST`, `MC_DIR_NEGATIVE` or `MC_DIR_CURRENT`)
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Commanded position finally reached
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification

Notes:

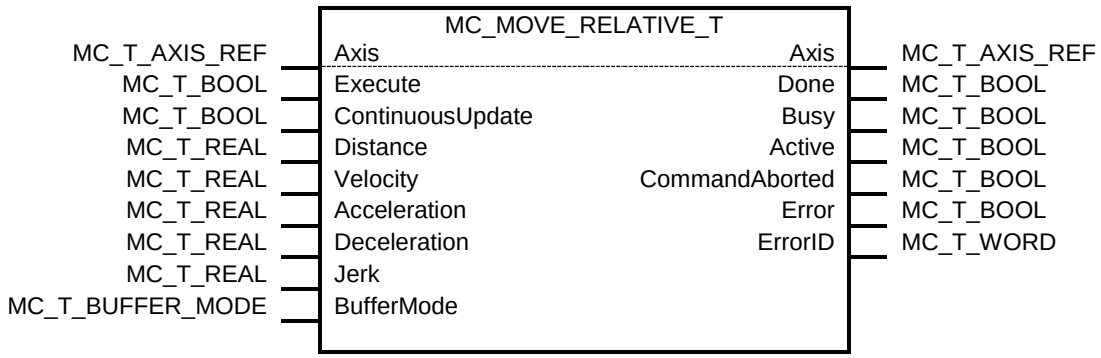
- This action completes with velocity zero if no further actions are pending
- If there is only one mathematical solution to reach the 'CommandedPosition' (like in linear systems), the value of the input 'Direction' is ignored
- For modulo axis - valid absolute position values are in the range of [0, 360[, (360 is excluded), or corresponding range. The application however may shift the 'CommandedPosition' of MC_MOVE_ABSOLUTE_T into the corresponding modulo range.
- The Enum type `MC_DIR_SHORTEST` is focused to a trajectory which will go through the shortest route. The decision which direction to go is based on the current position where the command is issued.



ContinuousUpdate and BufferMode are only supported in modes „cyclic synchronous position“ and “cyclic synchronous velocity”.

4.2.1.3 MC_MOVE_RELATIVE_T

class		**MC_MOVE_RELATIVE_T**		
This Function Block commands a controlled motion of a specified distance relative to the set position at the time of the execution.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start the motion at rising edge
	E	ContinuousUpdate	MC_T_BOOL	Trajectory is continuously updated.
	B	Distance	MC_T_REAL	Relative distance for the motion (in technical unit [u])
	E	Velocity	MC_T_REAL	Value of the maximum velocity (not necessarily reached) [u/s]
	E	Acceleration	MC_T_REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
	E	Deceleration	MC_T_REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
	E	Jerk	MC_T_REAL	Value of the Jerk [u/s^3]
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Commanded distance reached
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Notes: This action completes with velocity zero if no further actions are pending.				

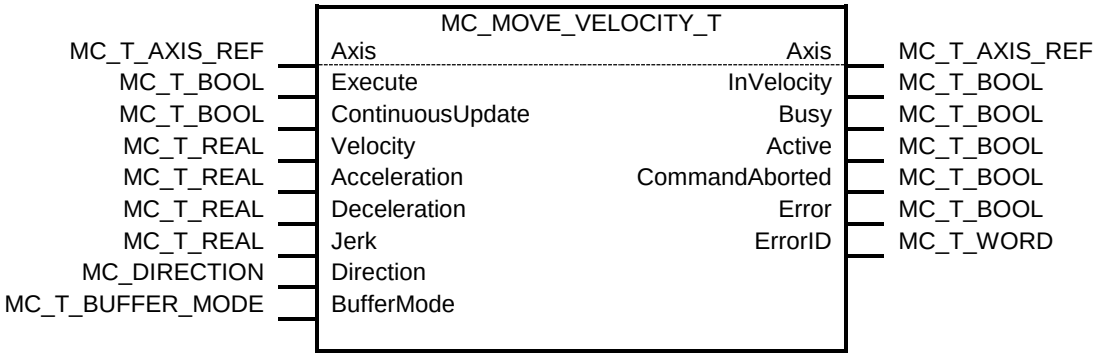


ContinuousUpdate and BufferMode are only supported in modes „cyclic synchronous position" and "cyclic synchronous velocity".

4.2.1.4 MC_MOVE_VELOCITY_T

class		**MC_MOVE_VELOCITY_T**		
This Function Block commands a never ending controlled motion at a specified velocity.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start the motion at rising edge
	E	ContinuousUpdate	MC_T_BOOL	Trajectory is continuously updated.
	B	Velocity	MC_T_REAL	Value of the maximum velocity [u/s]. Can be a signed value.
	E	Acceleration	MC_T_REAL	Value of the acceleration (increasing energy of the motor) [u/s^2]
	E	Deceleration	MC_T_REAL	Value of the deceleration (decreasing energy of the motor) [u/s^2]
	E	Jerk	MC_T_REAL	Value of the Jerk [u/s^3]
	E	Direction	MC_T_DIRECTION	Enum type (1-of-3 values: `MC_DIR_POSITIVE`, `MC_DIR_NEGATIVE` or `MC_DIR_CURRENT`). Note: shortest way not applicable.
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT				
	B	InVelocity	MC_T_BOOL	Commanded velocity reached
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification

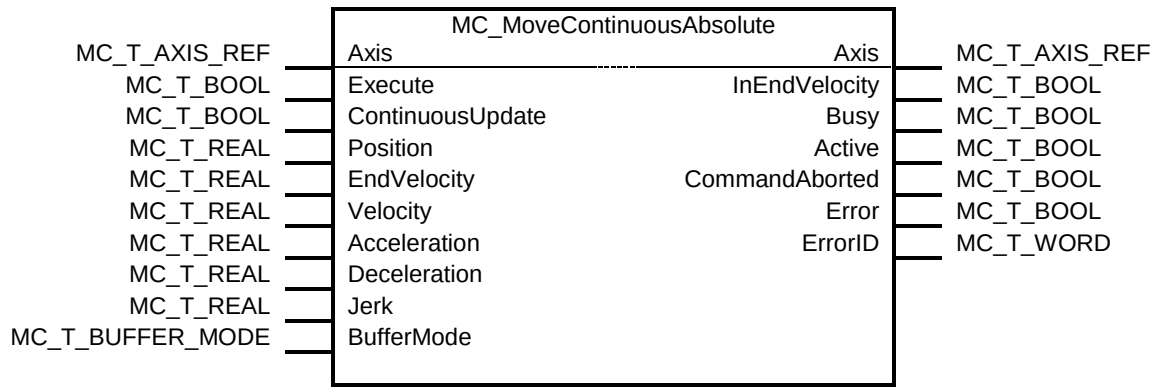
Notes:
- To stop the motion, the FB has to be interrupted by another FB issuing a new command
- The signal 'InVelocity' has to be reset when the block is aborted by another block.
- Negative velocity * negative direction = positive velocity
- In combination with MC_MoveSuperimposed, the output 'InVelocity' is SET as long as the contribution of this FB (MC_MOVE_VELOCITY_T) to the set velocity is equal to the commanded velocity of this FB.
- Not supported in PP mode



4.2.1.5 MC_MOVE_CONT_ABSOLUTE_T

Class		MC_MOVE_CONT_ABSOLUTE_T	
This Function Block commands a controlled motion to a specified absolute position ending with the specified velocity.			
VAR_IN_OUT			
B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT			
B	Execute	MC_T_BOOL	Start the motion at rising edge
E	ContinuousUpdate	MC_T_BOOL	See **Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.**'
B	Position	MC_T_REAL	Commanded position for the motion (in technical unit [u]) (negative or positive)
B	EndVelocity	MC_T_REAL	Value of the end velocity [u/s]. Signed value
B	Velocity	MC_T_REAL	Value of the maximum velocity [u/s]
E	Acceleration	MC_T_REAL	Value of the acceleration [u/s^2]
E	Deceleration	MC_T_REAL	Value of the deceleration [u/s^2]
E	Jerk	MC_T_REAL	Value of the Jerk [u/s^3]
E	Direction	MC_DIRECTION	Enum type (1-of-4 values: `MC_DIR_POSITIVE`, `MC_DIR_SHORTEST`, `MC_DIR_NEGATIVE` or `MC_DIR_CURRENT`)
E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT			
B	InEndVelocity	MC_T_BOOL	Commanded distance reached and running at requested end velocity
E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
B	ErrorID	MC_T_WORD	Error identification

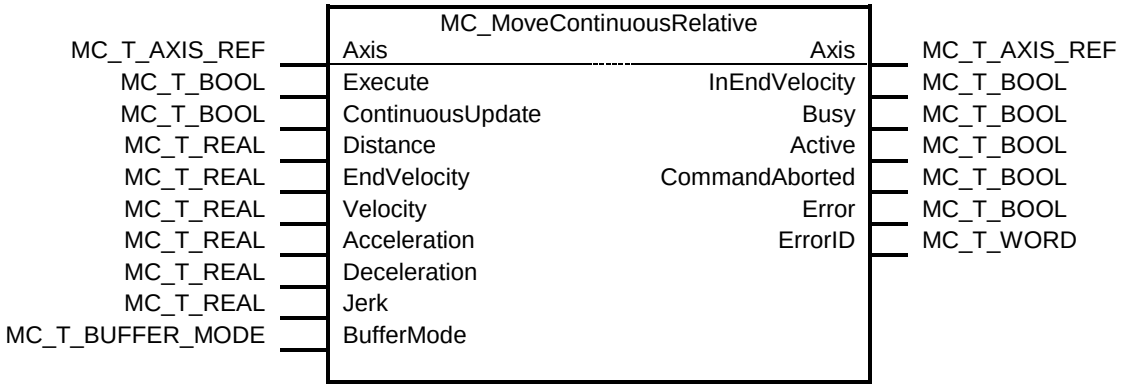
Notes:
- If the commanded position is reached and no new motion command is put into the buffer, the axis continues to run with the specified 'EndVelocity'.
- State 'ContinuousMotion' (meaning: it will not stop by itself).



4.2.1.6 MC_MOVE_CONT_RELATIVE_T

Class		MC_MOVE_CONT_RELATIVE_T	
This Function Block commands a controlled motion of a specified relative distance ending with the specified velocity.			
VAR_IN_OUT			
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT			
	B	Execute	MC_T_BOOL	Start the motion at rising edge
	E	ContinuousUpdate	MC_T_BOOL	See **Fehler! Verweisquelle konnte nicht gefunden werden. Fehler! Verweisquelle konnte nicht gefunden werden.**'
	B	Distance	MC_T_REAL	Relative distance for the motion [u]
	B	EndVelocity	MC_T_REAL	Value of the end velocity [u/s]. Signed value
	B	Velocity	MC_T_REAL	Value of the maximum velocity [u/s]
	E	Acceleration	MC_T_REAL	Value of the acceleration [u/s^2]
	E	Deceleration	MC_T_REAL	Value of the deceleration [u/s^2]
	E	Jerk	MC_T_REAL	Value of the Jerk [u/s^3]
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT			
	B	InEndVelocity	MC_T_BOOL	Commanded distance reached and running at requested end velocity
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	B	ErrorID	MC_T_WORD	Error identification

Notes:
- If the commanded position is reached and no new motion command is put into the buffer, the axis continues to run with the specified 'EndVelocity'.
- State 'ContinuousMotion' (meaning: it will not stop by itself).

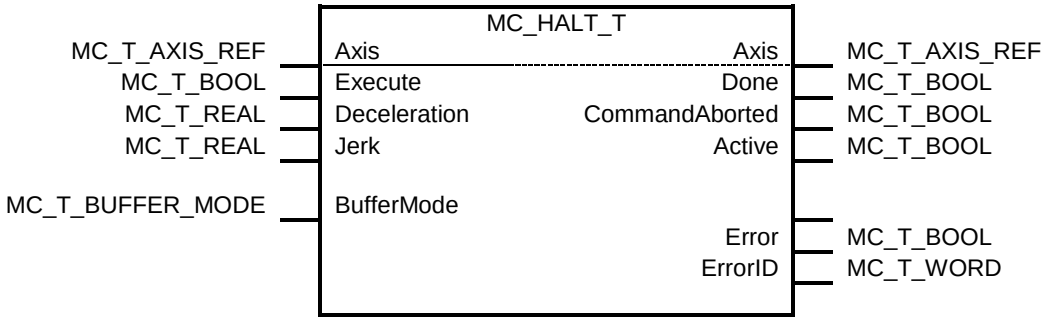


4.2.1.7 MC_HALT_T

class		MC_HALT_T		
This Function Block commands a controlled motion stop. The axis is moved to the state 'DiscreteMotion', until the velocity is zero. With the 'Done' output set, the state is transferred to 'Standstill'.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start the action at rising edge
	E	Deceleration	MC_T_REAL	Value of the 'Deceleration' [u/s^2]
	E	Jerk	MC_T_REAL	Value of the 'Jerk' [u/s^3]
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Zero velocity reached
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification

Notes:

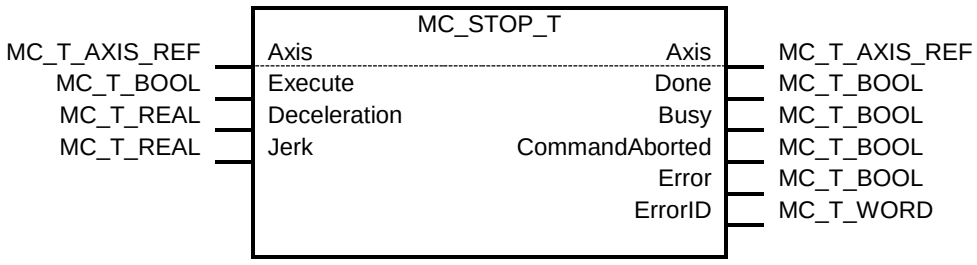
- MC_HALT_T is used to stop the axis under normal operation conditions. In non-buffered mode it is possible to set another motion command during deceleration of the axis, which will abort the MC_HALT_T and will be executed immediately.
- If this command is active the next command can be issued. E.g. a driverless vehicle detects an obstacle and needs to stop. MC_HALT_T is issued. Before the 'Standstill' is reached the obstacle is removed and the motion can be continued by setting another motion command, so the vehicle does not stop.



4.2.1.8 MC_STOP_T

Class		**MC_STOP_T**		
This Function Block commands a controlled motion stop and transfers the axis to the state ‘Stopping’. It aborts any ongoing Function Block execution. While the axis is in state ‘Stopping’, no other FB can perform any motion on the same axis. After the axis has reached ‘Velocity’ zero, the ‘Done’ output is set to TRUE immediately. The axis remains in the state ‘Stopping’ as long as ‘Execute’ is still TRUE or ‘Velocity’ zero is not yet reached. As soon as ‘Done’ is SET and ‘Execute’ is FALSE the axis goes to state ‘Standstill’.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start the action at rising edge
	E	Deceleration	MC_T_REAL	Value of the ‘Deceleration’ [u/s^2]
	E	Jerk	MC_T_REAL	Value of the ‘Jerk’ [u/s^3]
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Zero velocity reached
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	CommandAborted	MC_T_BOOL	‘Command’ is aborted by switching off power (only possibility to abort)
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification

Note:
1. This FB is primarily intended for emergency stop functionality or exception situations
2. As long as ‘Execute’ is high, the axis remains in the state ‘Stopping’ and may not be executing any other motion command.
3. If ‘Deceleration’ = 0, the behavior of the function block is implementation specific



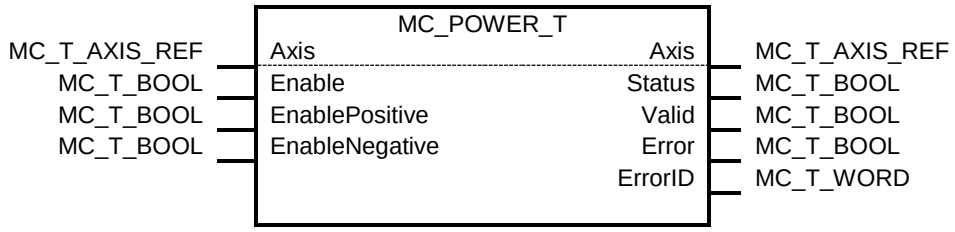
4.2.2 Administrative MCFB's

4.2.2.1 MC_POWER_T

Class		**MC_POWER_T**		
This Function Block controls the power stage (On or Off).				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	As long as 'Enable' is true, power is being enabled.
	E	EnablePositive	MC_T_BOOL	As long as 'Enable' is true, this permits motion in positive direction. *Note: Shall be set if Enable is set, but the functionality is not supported in the actual implementation.*
	E	EnableNegative	MC_T_BOOL	As long as 'Enable' is true, this permits motion in negative direction. *Note: Shall be set if Enable is set, but the functionality is not supported in the actual implementation.*
VAR_OUTPUT				
	B	Status	MC_T_BOOL	Effective state of the power stage
	E	Valid	MC_T_BOOL	If true, a valid set of outputs is available at the FB
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification

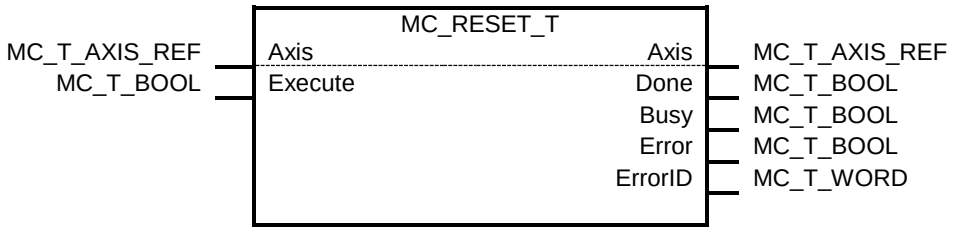
Notes:

- The 'Enable' input enables the power stage in the drive and not the FB itself
- If the MC_POWER_T FB is called with the 'Enable' = TRUE while being in 'Disabled', the axis state changes to 'Standstill'.
- It is possible to set an error variable when the Command is TRUE for a while and the Status remains false with a Timer FB and an AND Function (with inverted Status input). It indicates that there is a hardware problem with the power stage.
- If power fails (also during operation) it will generate a transition to the 'ErrorStop' state.
- 'EnablePositive' and 'EnableNegative' are both level sensitive.
- 'EnablePositive' & 'EnableNegative' can both be true.
- Only 1 FB MC_POWER_T should be issued per axis.



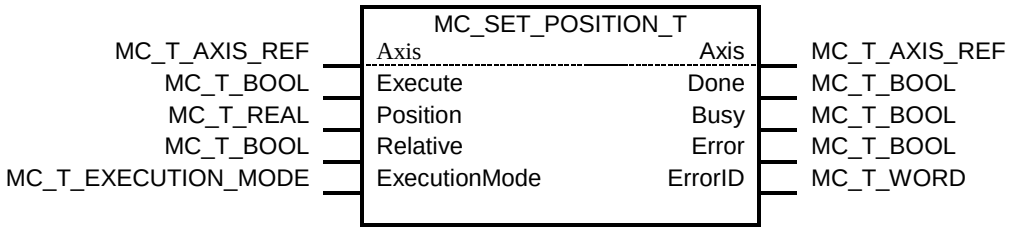
4.2.2.2 MC_RESET_T

Class		**MC_RESET_T**		
This Function Block makes the transition from the state ‘ErrorStop’ to ‘Standstill’ or ‘Disabled’ by resetting all internal axis-related errors – it does not affect the output of the FB instances.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Resets all internal axis-related errors
VAR_OUTPUT				
	B	Done	MC_T_BOOL	‘Standstill’ or ‘Disabled’ state is reached
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Note: the application of MC_RESET_T in other states then the state ‘ErrorStop’ is vendor specific				



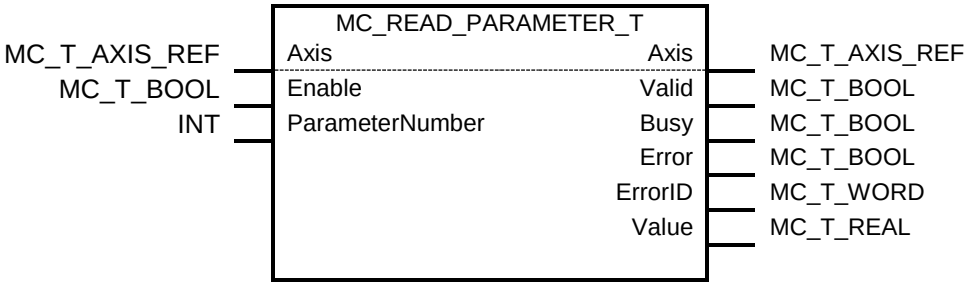
4.2.2.3 MC_SET_POSITION_T

Class		MC_SET_POSITION_T	
This Function Block shifts the coordinate system of an axis by manipulating both the set-point position as well as the actual position of an axis with the same value without any movement caused. (Re-calibration with same following error). This can be used for instance for a reference situation. This Function Block can also be used during motion without changing the commanded position, which is now positioned in the shifted coordinate system.			
VAR_IN_OUT			
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT			
	B	Execute	MC_T_BOOL	Start setting position in axis
	B	Position	MC_T_REAL	Position unit [u] (Means ‘Distance’ if ‘Relative’= TRUE)
	E	Relative	MC_T_BOOL	‘Relative’ distance if True, ‘Absolute’ position if False (= Default)
	E	ExecutionMode	MC_T_EXECUTION_MODE	ENUM. Defines the chronological sequence of the FB. `MC_IMMEDIATELY` - the functionality is immediately valid and may influence the on-going motion but not the state (note: is the default behaviour) `MC_QUEUED` – Not supported.
VAR_OUTPUT			
	B	Done	MC_T_BOOL	‘Position’ has new value
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Note: ‘Relative’ means that ‘Position’ is added to the actual position value of the axis at the time of execution. This results in a recalibration by a specified distance. ‘Absolute’ means that the actual position value of the axis is set to the value specified in the ‘Position’ parameter.			



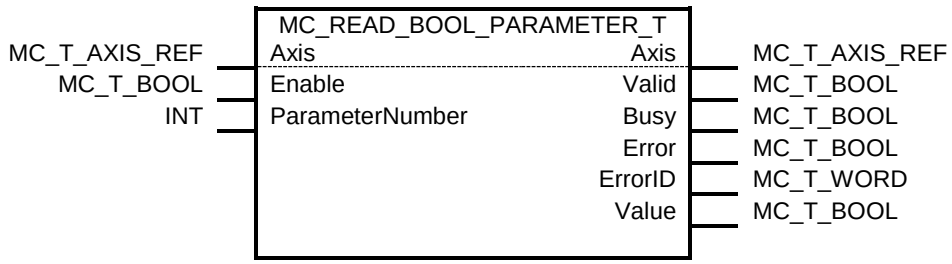
4.2.2.4 MC_READ_PARAMETER_T

Class		MC_READ_PARAMETER_T		
This Function Block returns the value of a vendor specific parameter. The returned Value has to be converted to MC_T_REAL if necessary. If not possible, the vendor has to supply a vendor specific FB to read the parameter.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
	B	ParameterNumber	INT	Number of the parameter
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	B	Value	MC_T_REAL	Value of the specified parameter in the datatype, as specified by the vendor
Note: The parameters are defined in the table below.				



4.2.2.5 MC_READ_BOOL_PARAMETER_T

Class		**MC_READ_BOOL_PARAMETER_T**		
This Function Block returns the value of a vendor specific parameter with datatype MC_T_BOOL.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
	B	ParameterNumber	INT	Number of the parameter
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function block
	E	ErrorID	MC_T_WORD	Error identification
	B	Value	MC_T_BOOL	Value of the specified parameter in the datatype, as specified by the vendor
Note: The parameters are defined in the table below				



These parameters are available for use in the application program, and typically are not intended for commissioning tools like operator panels, etc. (the drive is not visible – only the axis position)
Note: that the most used parameters are accessible via Function Blocks, and are not listed here.

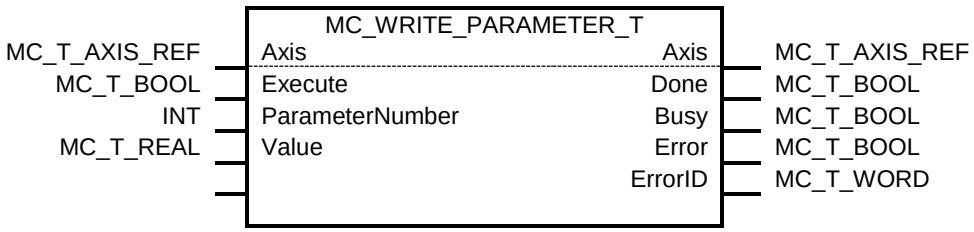
Note: PN is Parameter Number

PN	**Name**	**Datatype**	**B/E**	**R/W**	**Comments**
1	MCFB_PN_COMMANDED_POSITION	MC_T_REAL	B	R	Commanded position
2	MCFB_PN_SW_LIMIT_POS	MC_T_REAL	E	R/W	Positive Software limit switch position
3	MCFB_PN_SW_LIMIT_NEG	MC_T_REAL	E	R/W	Negative Software limit switch position
4	MCFB_PN_ENA_LIMIT_POS	MC_T_BOOL	E	R/W	Enable positive software limit switch
5	MCFB_PN_ENA_LIMIT_NEG	MC_T_BOOL	E	R/W	Enable negative software limit switch
10	MCFB_PN_ACTUAL_VELOCITY	MC_T_REAL	B	R	Actual velocity
11	MCFB_PN_COMMANDED_VELOCITY	MC_T_REAL	B	R	Commanded velocity
1000	MCFB_PN_COMMANDED_ACCELERATION	MC_T_REAL	V	R	Commanded acceleration
1001	MCFB_PN_COMMANDED_JERK	MC_T_REAL	V	R	Commanded jerk

Table 1: Parameters for MC_READ_PARAMETER_T and MC_WRITE_PARAMETER_T

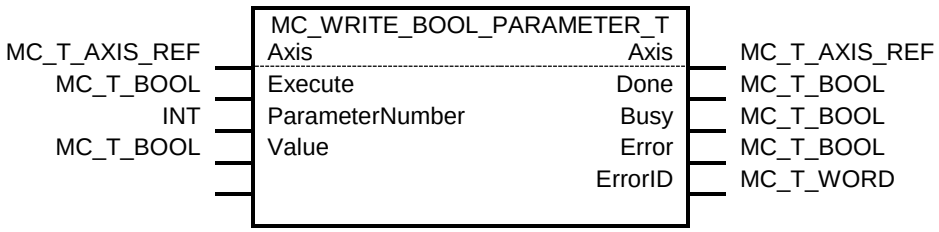
4.2.2.6 MC_WRITE_PARAMETER_T

Class		**MC_WRITE_PARAMETER_T**		
This Function Block modifies the value of a vendor specific parameter.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Write the value of the parameter at rising edge
	B	ParameterNumber	INT	Number of the parameter (correspondence between number and parameter is specified in the table above)
	B	Value	MC_T_REAL	New value of the specified parameter
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Parameter successfully written
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected.
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Notes: The parameters are defined in the table above (under MC_READ_PARAMETER_T, writing allowed)				



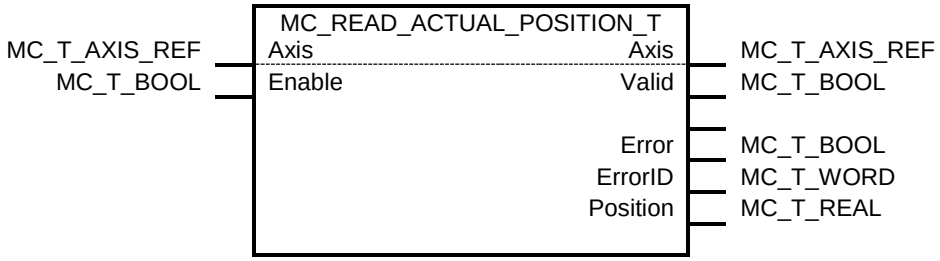
4.2.2.7 MC_WRITE_BOOL_PARAMETER_T

Class			**MC_WRITE_BOOL_PARAMETER_T**	
This Function Block modifies the value of a vendor specific parameter of type MC_T_BOOL.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Write the value of the parameter at rising edge
	B	ParameterNumber	INT	Number of the parameter (correspondence between number and parameter is specified in the table above)
	B	Value	MC_T_BOOL	New value of the specified parameter
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Parameter successfully written
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected.
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Notes: The parameters are defined in the table above (under MC_READ_PARAMETER_T, writing allowed)				



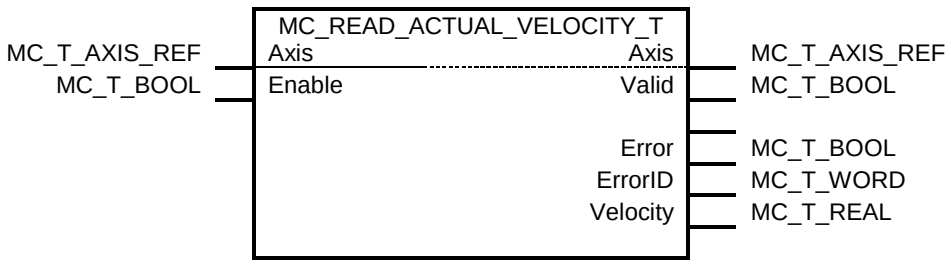
4.2.2.8 MC_READ_ACTUAL_POSITION_T

Class		**MC_READ_ACTUAL_POSITION_T**		
This Function Block returns the actual position.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	B	Position	MC_T_REAL	New absolute position (in axis' unit [u])



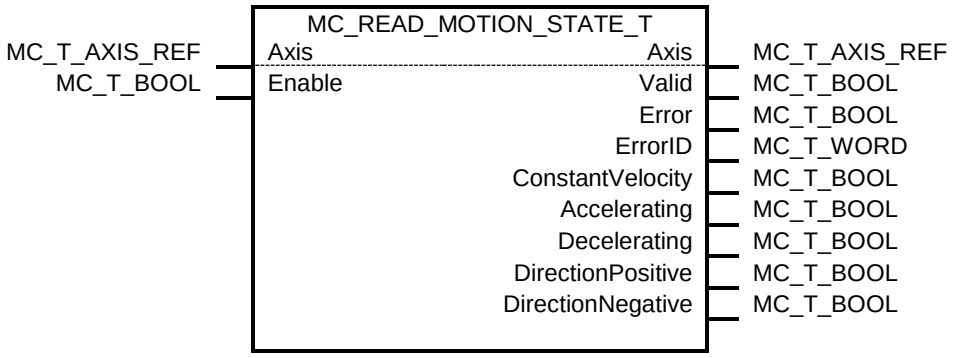
4.2.2.9 MC_READ_ACTUAL_VELOCITY_T

Class			**MC_READ_ACTUAL_VELOCITY_T**	
This Function Block returns the value of the actual velocity as long as ‘Enable’ is set. ‘Valid’ is true when the data-output ‘Velocity’ is valid. If ‘Enable’ is Reset, the data loses its validity, and all outputs are reset, no matter if new data is available.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	B	Velocity	MC_T_REAL	The value of the actual velocity (in axis’ unit [u/s])
Notes: The output ‘Velocity’ can be a signed value				



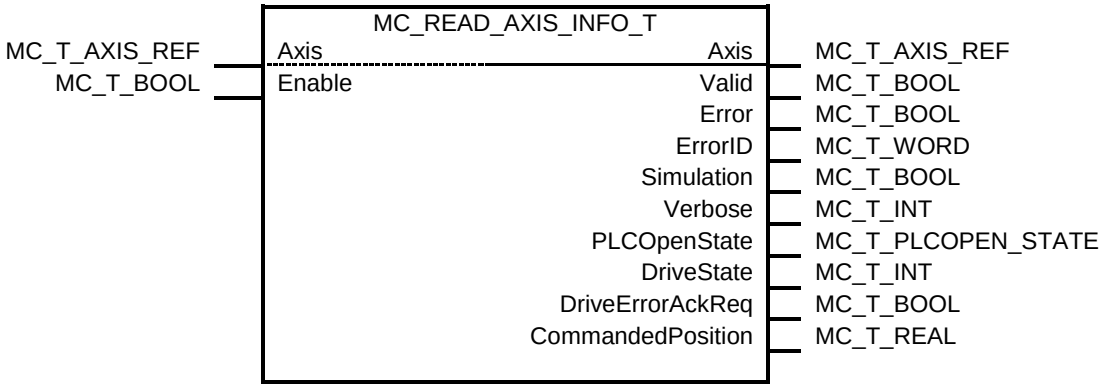
4.2.2.10 MC_READ_MOTION_STATE_T

Class			**MC_READ_MOTION_STATE_T**	
This Function Block returns in detail the status of the axis with respect to the motion currently in progress.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	True if a valid set of outputs available
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function block
	E	ErrorID	MC_T_WORD	Error identification
	E	ConstantVelocity	MC_T_BOOL	Velocity is constant. Velocity may be 0. For the actual value a window is applicable (window is vendor specific)
	E	Accelerating	MC_T_BOOL	Increasing the absolute value of the velocity
	E	Decelerating	MC_T_BOOL	Decreasing the absolute value of the velocity
	E	DirectionPositive	MC_T_BOOL	Signals that the position is increasing
	E	DirectionNegative	MC_T_BOOL	Signals that the position is decreasing



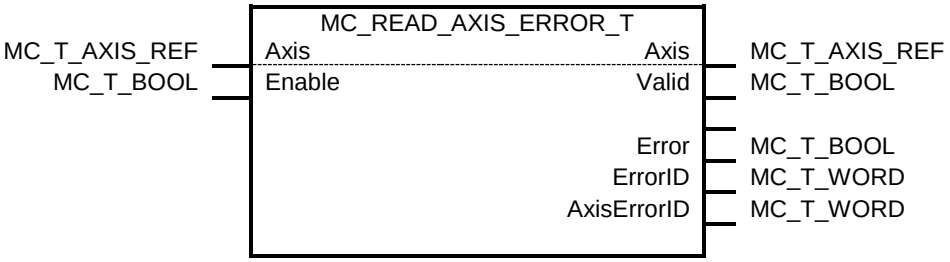
4.2.2.11 MC_READ_AXIS_INFO_T

Class		**MC_READ_AXIS_INFO_T**		
This Function Block reads information concerning an axis, like modes, inputs directly related to the axis, and certain status information.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the axis information constantly while enabled
	V	DriveErrorAck	MC_T_BOOL	Set by user if a drive error has been acknowledged. See also DriveErrorAckReq.
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	True if a valid set of outputs is available
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	E	Simulation	MC_T_BOOL	Axis is in simulation mode (e.g. motor is simulated)
	V	Verbose	MC_T_INT	Verbosity level for diagnosis output.
	V	PLCOpenState	MC_T_PLCOPEN_STATE	Current state of the PLCOpen state machine. This is intended for diagnosis purposes only.
	V	DriveState	MC_T_INT	CiA 402 / SERCOS / SIMU drive state. This is intended for diagnosis purposes only and is subject to change.
	V	DriveErrorAckReq	MC_T_BOOL	Set by motion kernel if a drive error should be acknowledged by the user.
	V	CommandedPosition	MC_T_REAL	Read commanded position.



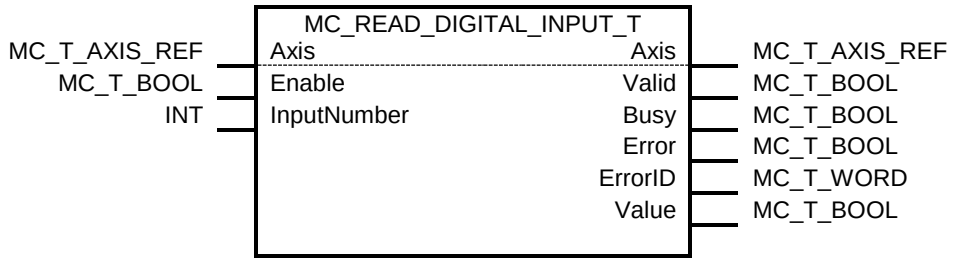
4.2.2.12 MC_READ_AXIS_ERROR_T

Class		**MC_READ_AXIS_ERROR_T**	
This Function Block presents general axis errors not relating to the Function Blocks. (for instance axis errors, drive errors, communication errors)			
VAR_IN_OUT			
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT			
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
VAR_OUTPUT			
	B	Valid	MC_T_BOOL	True if a valid output is available at the FB
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	B	ErrorID	MC_T_WORD	Error identification
	E	AxisErrorID	MC_T_WORD	The value of the axis error. These values are vendor specific
Notes: -			



4.2.2.13 MC_READ_DIGITAL_INPUT_T

Class		MC_READ_DIGITAL_INPUT_T		
This Function Block gives access to the value of the input, referenced by the datatype MC_T_AXIS_REF. It provides the value of the referenced input (BOOL)				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
	B	InputNumber	INT	Selects the input.
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	B	Value	MC_T_BOOL	The value of the selected input signal
Note: It is not guaranteed that the digital signal will be seen by the FB: a short pulse on the digital input could be over before the next Function Block cycle occurs.				



Important: Digital inputs have to be mapped into process data and a valid pointer has to be assigned to MC_T_AXIS_INIT_INPUTS::pDigitalInputs member.

Comments

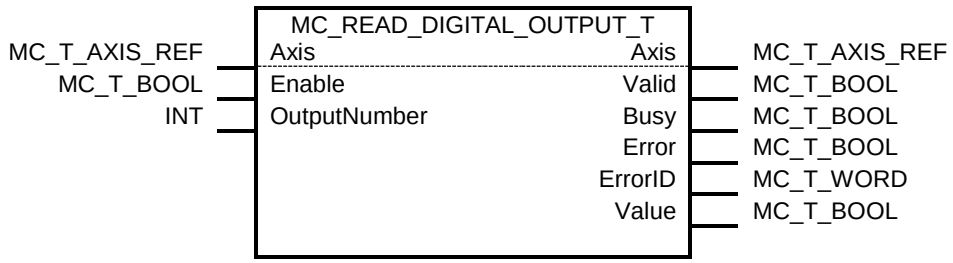
For an axis conform to CiA402, the object “Digital Inputs” Index 0x60FD, Subindex 0 has to be mapped into a Transmit PDO.
The pointer to the “Digital Inputs” in the input process image has to be assigned to MC_T_AXIS_INIT_INPUTS.pDigitalInputs. See also InitInputs(const MC_T_AXIS_INIT_INPUTS &)

Defines

```
#define DRV_OBJ_DIGITAL_INPUTS              0x60FD
struct MC_T_AXIS_INIT_INPUTS
class _MC_API MC_READ_DIGITAL_INPUT_T
```

4.2.2.14 MC_READ_DIGITAL_OUTPUT_T

Class		MC_READ_DIGITAL_OUTPUT_T		
This Function Block provides access to the value of a digital output, referenced by the datatype MC_T_AXIS_REF. It provides the value of the referenced output (BOOL).				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
	B	OutputNumber	INT	Selects the output.
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	B	Value	MC_T_BOOL	The value of the selected output signal
Note: It is not guaranteed that the digital signal will be seen by the FB: a short pulse on the digital output could be over before the next Function Block cycle occurs.				



Important: Digital outputs have to be mapped into process data and a valid pointer has to be assigned to MC_T_AXIS_INIT_OUTPUTS::pDigitalOutputs member.

Comments

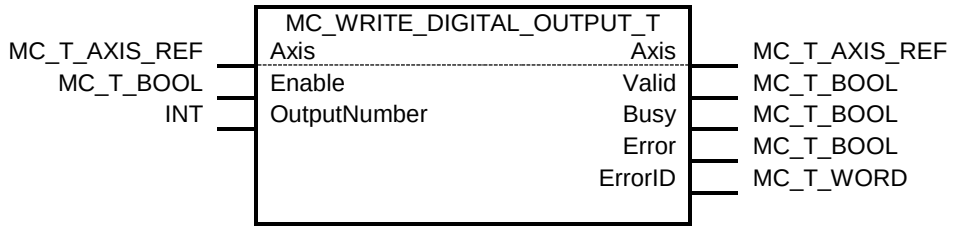
For an axis conform to CiA402, the object “Digital Outputs” Index 0x60FE, Subindex 1 has to be mapped into a Receive PDO.
The pointer to the “Digital Outputs” in the output process image has to be assigned to MC_T_AXIS_INIT_OUTPUTS.pDigitalOutputs. See also InitOutputs(const MC_T_AXIS_INIT_OUTPUTS &)

Defines

```
#define DRV_OBJ_DIGITAL_OUTPUTS             0x60FE
struct MC_T_AXIS_INIT_OUTPUTS
class _MC_API MC_READ_DIGITAL_OUTPUT_T
```

4.2.2.15 MC_WRITE_DIGITAL_OUTPUT_T

Class		MC_WRITE_DIGITAL_OUTPUT_T		
This Function Block writes a value to the output referenced by the argument 'Axis' once (with rising edge of Execute).				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Enable	MC_T_BOOL	Get the value of the parameter continuously while enabled
	B	OutputNumber	INT	Selects the output.
VAR_OUTPUT				
	B	Valid	MC_T_BOOL	A valid output is available at the FB
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Note: -				



Important: Digital outputs have to be mapped into process data and a valid pointer has to be assigned to MC_T_AXIS_INIT_OUTPUTS::pDigitalOutputs member.

Comments

For an axis conform to CiA402, the object "Digital Outputs" Index 0x60FE, Subindex 1 has to be mapped into a Receive PDO.
The pointer to the "Digital Outputs" in the output process image has to be assigned to MC_T_AXIS_INIT_OUTPUTS.pDigitalOutputs. See also InitOutputs(const MC_T_AXIS_INIT_OUTPUTS &)

Defines

```
#define DRV_OBJ_DIGITAL_OUTPUTS             0x60FE
struct MC_T_AXIS_INIT_OUTPUTS
class _MC_API MC_WRITE_DIGITAL_OUTPUT_T
```

4.3 Camming Function Blocks

4.3.1 CAM Table

In the CAM table the CAM profile is defined. Currently in CAM table the master and slave positions have to be defined within two dimensional vector.

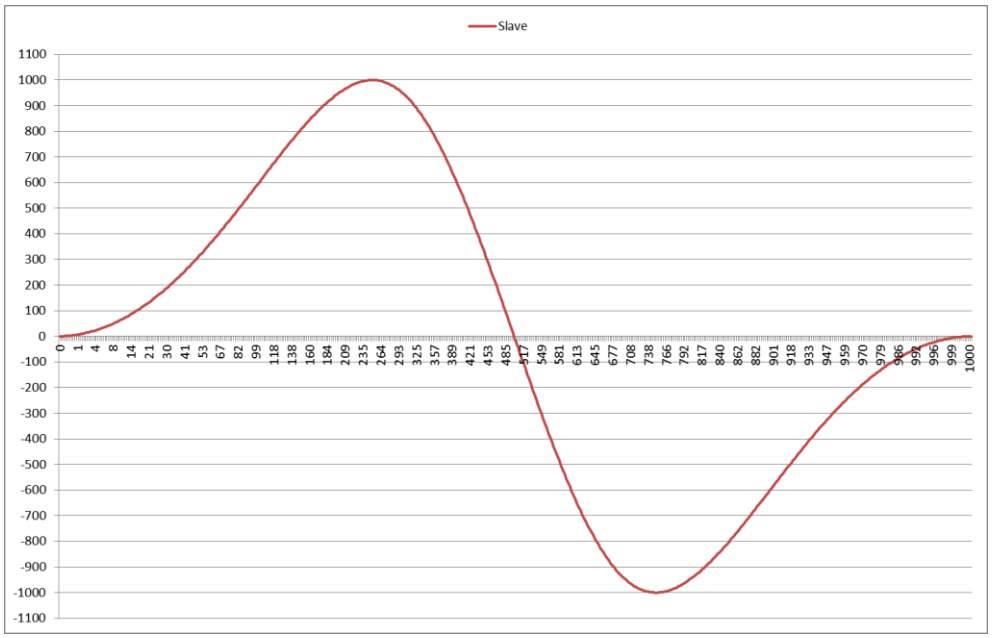
```
enum MC_T_CAM_VAR_TYPE
{
   MC_CAM_VAR_TYPE_INT       = 1,      /* table contains MC_T_INT values  */
   MC_CAM_VAR_TYPE_REAL      = 2       /* table contains MC_T_REAL values */
};

enum MC_T_CAM_INTERPOL_TYPE
{
   MC_CAM_INTERPOL_TYPE_LIN  = 1,      /* interpolation type linear */
   MC_CAM_INTERPOL_TYPE_CUB  = 2       /* interpolation type cubic  */
};

typedef struct
{
   MC_T_CAM_VAR_TYPE    eVarType;          /* variable type      */
   MC_T_CAM_INTERPOL_TYPE eInterpolType;   /* interpolation type */
   MC_T_INT             nNumOfElements;    /* number of elements */
   EC_T_VOID*           pData;             /* two dimensional table with master/slave positions */
} MC_T_CAM_REF;
```

Comments

The minimum number of elements (points) is 2.
The maximum number of elements (points) support is defined by *MAX_NUM_OF_SPLINE_POINTS*.

Example

```
static MC_T_CAM_REF S_myCamTable;
static MC_T_INT     S_aCamPointsInt[][2] = {{0, 0}, {125,707}, {250,1000}, {375,707},\
{500,0}, {625,-707}, {750,-1000}, {875,-707}, {1000,0}};

S_myCamTable.eInterpolType = MC_CAM_INTERPOL_TYPE_CUB;
S_myCamTable.nNumOfElements = sizeof(S_aCamPointsInt)/2/sizeof(MC_T_INT);
S_myCamTable.eVarType = MC_CAM_VAR_TYPE_INT;
S_myCamTable.pData = S_aCamPointsInt;
```



4.3.2 Calculation of Slave axis position

4.3.2.1 Formula

```
/* Calculate MasterPos considering offsets and scaling */
MasterPos = ((MasterPosPhysical - MasterStartOffset) * MasterScaling) + MasterOffset;

/* Get SlavePos from spline (CAM profile */
SlavePos  = Spline->get_result(MasterPos);

/* Calculate SlavePos considering offsets and scaling */
SlavePos  = (SlavePos * SlaveScaling) + SlaveOffset + SlaveStartOffset;
```

This formula is implemented in MC_T_CAM_ID::CalcSlavePos().

4.3.2.2 Engage procedure

The engage procedure is activated by the function block MC_CAM_IN_T.

Prerequisites for engaging are:

- Master and Slave axis are powered on
- Provide a valid CAM table reference generated by MC_CAMTABLE_SELECT_T
- Master axis state is either MC_PLCOPEN_STATE_STAND_STILL, MC_PLCOPEN_STATE_DISC_MOTION or MC_PLCOPEN_STATE_CONT_MOTION
- Slave axis state is MC_PLCOPEN_STATE_STAND_STILL
- If ´MasterAbsolute=MC_TRUE´, the actual master position has to be inside the range defined in the CAM table. This check is also considering ´MasterScaling´ and ´MasterOffset´
- If ´MasterAbsolute=MC_FALSE´, the first master position in the CAM table has to be zero
- If ´StartMode=MC_SM_RAMP_IN´, the parameters ´Velocity´, ´ Acceleration´ and ´Deceleration´ are required

After the engage procedure is successful the slave axis is in state *MC_PLCOPEN_STATE_SYNC_MOTION*.

Important parameters

- MC_CAMTABLE_SELECT_T.MasterAbsolute:
  If true, the CAM table master positions are used as absolute positions. If false, the CAM table starts at the actual master position (MasterStartOffset is set to actual position).
- MC_CAMTABLE_SELECT_T.SlaveAbsolute:
  If true, the CAM table slave positions are used as absolute positions. If false, the CAM table starts at the actual slave position (SlaveStartOffset is set to actual position).
- MC_CAM_IN_T.MasterOffset:
  Additional offset on master position
- MC_CAM_IN_T.SlaveOffset:
  Additional offset on slave position
- MC_CAM_IN_T.StartMode
  MC_SM_ABSOLUTE Sync immediately: Slave axis may jump to position defined in CAM profile
  MC_SM_RELATIVE Actual slave position is used as SlaveStartOffset. ´SlaveAbsolute´ has no impact.
  MC_SM_RAMP_IN Slave axis will move to position defined in CAM profile based on the actual master position. The movement to this position is calculated based on the parameters ´Velocity´, ´ Acceleration´ and ´Deceleration´. The master axis has to stand still.

4.3.2.3 Camming in Operation

To move the engaged slave axis based on the CAM profile, the master axis has to be moved, e. g. with MC_MOVE_ABSOLUTE_T, MC_MOVE_VELOCITY_T.

Important parameters

- MC_CAM_IN_T.MasterScaling:
  Factor for the master axis profile. From the slave point of view the master overall profile is multiplied by this factor. If MasterScaling > 1 the CAM profile will be processed faster.
- MC_CAM_IN_T.SlaveScaling:
  Factor for the slave profile (default = 1.0). The overall slave profile is multiplied by this factor. If SlaveScaling > 1 the movement of the slave is bigger than defined in the CAM table.
- MC_CAMTABLE_SELECT_T.Periodic:
  If false and the master position is outside of the CAM profile, the slave axis stays in synchronized motion and keeps the last position. The CAM profile is run only once. In reverse mode, the CAM profile is not executed after having reached the 'EndOfProfile' position.
  If true, after the master position has reached the end of the profile (MasterEnd) the whole profile is shifted. In reserve direction the profile is shifted when the master position had reached the start of the profile (MasterStart).
  Forward Dir (+): `MasterStartOffset = MasterStartOffset + MasterPeriod;`
  Reserve Dir (-): `MasterStartOffset = MasterStartOffset - MasterPeriod;`

4.3.2.4 Disengage procedure

The disengage procedure is activated by the function block MC_CAM_OUT_T. Divergent to the PLCopen standard the slave axis will be stopped after disengaging from master axis. The parameters ´Deceleration´ and ´Jerk´ are used to calculate the break ramp.
After disengage the PLCOpenState of the slave axis is set to *MC_PLCOPEN_STATE_DISC_MOTION* and after the axis is stopped to *MC_PLCOPEN_STATE_STAND_STILL.*
The PLCOpenState of the master axis isn´t changed.

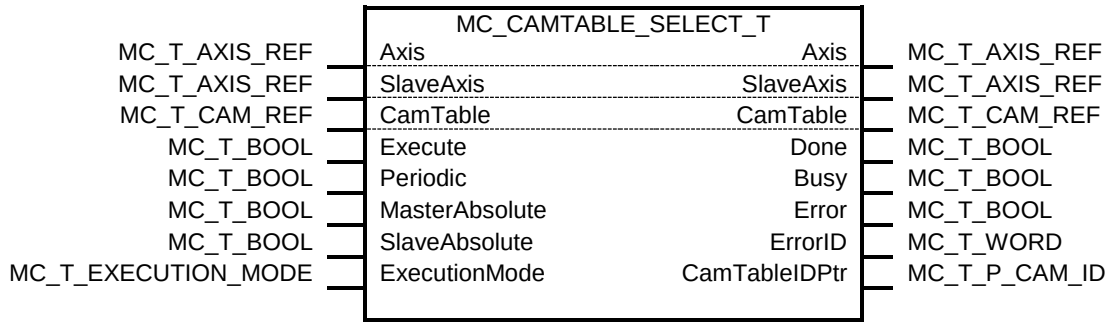
Important parameters

- Deceleration: Value of the deceleration (always positive) used to halt slave axis after disengage from master axis
- Jerk: Value of the jerk (always positive) used to halt slave axis after disengage from master axis

4.3.3 MC_CAMTABLE_SELECT_T

Class		**MC_CAMTABLE_SELECT_T**		
This Function Block selects the CAM tables by setting the connections to the relevant tables				
VAR_IN_OUT				
	E	Axis	MC_T_AXIS_REF	Reference to the master axis
	E	SlaveAxis	MC_T_AXIS_REF	Reference to the slave axis
	B	CamTable	MC_T_CAM_REF	Reference to CAM description
VAR_INPUT				
	B	Execute	MC_T_BOOL	Selection at rising edge
	E	Periodic	MC_T_BOOL	1 = periodic, 0 = non periodic (single-shot)
	E	MasterAbsolute	MC_T_BOOL	1 = absolute; 0 = relative coordinates
	E	SlaveAbsolute	MC_T_BOOL	1 = absolute; 0 = relative coordinates
	E	ExecutionMode	MC_T_EXECUTION_MODE	Not supported
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Pre-selection done
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	E	CamTableIDPtr	MC_T_P_CAM_ID	Identifier of CAM Table to be used in the MC_CAM_IN_T FB

Notes:
- A virtual axis can be used as master axis
- When the Done output is SET, the CamTableIDPtr is valid and ready for use in a MC_CAM_IN_T function block.



Return
MC_NO_ERROR if successful.
MC_ERR_SLAVE_AXIS_INVALID
MC_ERR_CAM_TABLE_ID_INVALID
MC_ERR_CAM_TABLE_ELEM_TOO_LESS
MC_ERR_CAM_TABLE_ELEM_TOO_MANY
MC_ERR_CAM_TABLE_VARTYPE_INVALID
MC_ERR_CAM_SPLINE_INVALID
MC_ERR_CAM_TABLE_DATA_INVALID
MC_ERR_CAM_IN_MAS_PERIOD_ZERO
MC_ERR_CAM_IN_MAS_PERIOD_ZERO
MC_ERR_CAM_TABLE_INTERPOL_INVALID

Example

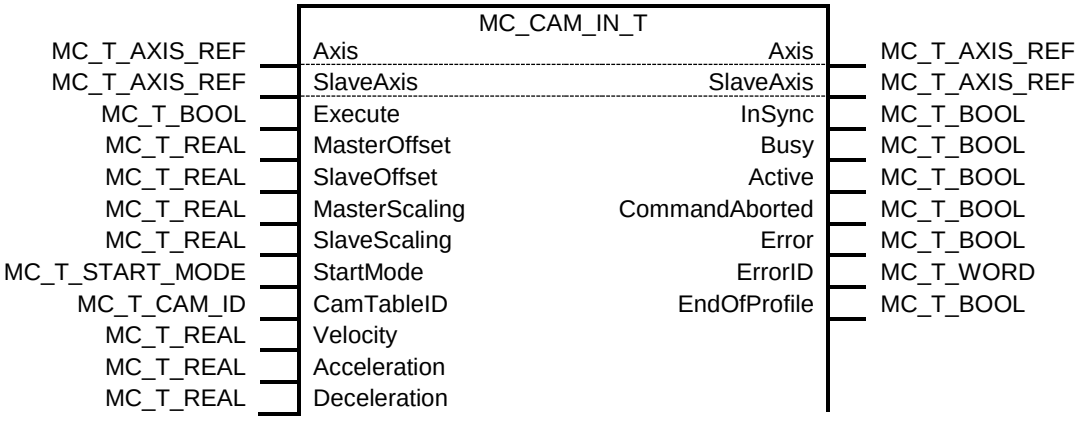
```
MC_CAMTABLE_SELECT_T  CamTableSelect;
MC_T_CAM_REF          S_myCamTable;
MC_T_CAM_ID*          S_myCamTableID;

CamTableSelect.Axis = pAxis;
CamTableSelect.SlaveAxis       = &pSlaveAxis->pFb->Axis;
CamTableSelect.CamTable        = &S_myCamTable;
CamTableSelect.Periodic        = MC_TRUE;
CamTableSelect.MasterAbsolute  = MC_FALSE;
CamTableSelect.SlaveAbsolute   = MC_TRUE;
CamTableSelect.Execute = MC_TRUE;

CamTableSelect.OnCycle();

if (CamTableSelect.Done)    /* done ? */
{
   S_myCamTableID = CamTableSelect.CamTableIDPtr;       /* save reference to CAM table */
   CamTableSelect.Execute = MC_FALSE;
}
```

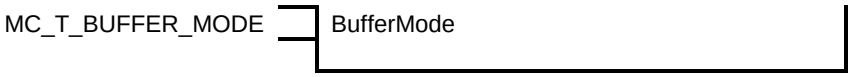
Important definitions

```
class _MC_API MC_CAMTABLE_SELECT_T
typedef struct{…} MC_T_CAM_REF;
typedef struct{…} MC_T_CAM_ID, *MC_T_P_CAM_ID;
```

4.3.4 MC_CAM_IN_T

Class		**MC_CAM_IN_T**		
This Function Block engages the CAM				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the master axis
	B	SlaveAxis	MC_T_AXIS_REF	Reference to the slave axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start at rising edge
	E	MasterOffset	MC_T_REAL	Offset of the master shaft to cam.
	E	SlaveOffset	MC_T_REAL	Offset of slave table.
	E	MasterScaling	MC_T_REAL	Factor for the master profile (default = 1.0). From the slave point of view the master overall profile is multiplied by this factor
	E	SlaveScaling	MC_T_REAL	Factor for the slave profile (default = 1.0). The overall slave profile is multiplied by this factor.
	E	StartMode	MC_T_START_MODE	Start mode: MC_SM_ABSOLUTE, MC_SM_RELATIVE, or MC_SM_RAMP_IN
	E	CamTableID	MC_T_CAM_ID	Identifier of CAM Table to be used, linked to output of MC_CAMTABLE_SELECT_T
	E	Velocity	MC_T_REAL	Only for StartMode=MC_SM_RAMP_IN: Value of the maximum velocity (always positive)
	E	Acceleration	MC_T_REAL	Only for StartMode=MC_SM_RAMP_IN: Value of the acceleration (always positive)
	E	Deceleration	MC_T_REAL	Only for StartMode=MC_SM_RAMP_IN: Value of the deceleration (always positive)
	E	BufferMode	MC_T_BUFFER_MODE	Defines the chronological sequence of the FB.
VAR_OUTPUT				
	B	InSync	MC_T_BOOL	Is TRUE if the set value = the commanded value.
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	E	Active	MC_T_BOOL	Indicates that the FB has control on the axis
	E	CommandAborted	MC_T_BOOL	'Command' is aborted by another command
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
	E	EndOfProfile	MC_T_BOOL	Pulsed output signaling the cyclic end of the CAM Profile It is displayed every time the end of the cam profile is reached. In reverse direction, the 'EndOfProfile' is displayed also at the end of the cam profile (in this case the first point of the cam profile)





Return

MC_NO_ERROR if successful.
MC_ERR_SLAVE_AXIS_INVALID
MC_ERR_CAM_TABLE_ID_INVALID
MC_ERR_START_MODE_INVALID
MC_ERR_INVALID_PLCOPEN_STATE
MC_ERR_INVALID_PLCOPEN_STATE_SLAVE
MC_ERR_CAM_IN_MASPOS_TOO_SMALL
MC_ERR_CAM_IN_MASPOS_TOO_BIG
MC_ERR_CAM_IN_MASPOS_NULL_MISSING

Comments

- If the actual master and slave positions do not correspond to the offset values when MC_CAM_IN_T is executed, either an error occurs or the system deals with the difference automatically
- The Cam is placed either absolute or relative to the current master and slave positions.
  Absolute: the profile between master and slave is seen as an absolute relationship.
  Relative: the relationship between master and slave is in a relative mode.
- Ramp-in is a supplier specific mode. It can be coupled to additional parameters, such as a master-distance parameter, acceleration parameter, or other supplier specific parameters where the slave to ramp-in into the cam profile (“ flying coupling”)

Example

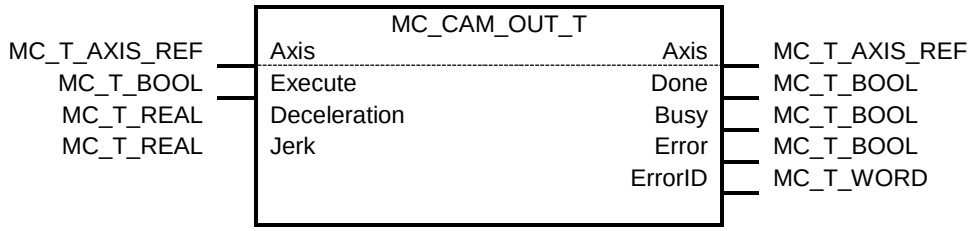
```
MC_CAM_IN_T  CamIn;

CamIn.Axis = pAxis;
CamIn.SlaveAxis   = &pSlaveAxis->pFb->Axis;
CamIn.StartMode   = MC_SM_RELATIVE;
CamIn.CamTableID  = S_myCamTableID;

CamIn.Execute = MC_TRUE;
CamIn.OnCycle();
```

Important definitions

```
class _MC_API MC_CAM_IN_T
typedef struct{…} MC_T_CAM_ID, *MC_T_P_CAM_ID;
```

4.3.5 MC_CAM_OUT_T

Class		**MC_CAM_OUT_T**	
This Function Block disengages the Slave axis from the Master axis immediately. The Slave axis will be stopped if moving.			
VAR_IN_OUT			
B	Axis	MC_T_AXIS_REF	Reference to the slave axis
VAR_INPUT			
B	Execute	MC_T_BOOL	Start to disengage the slave from the master
B	Deceleration	MC_T_REAL	Value of the deceleration (always positive) used to halt slave axis after disengage from master axis
B	Jerk	MC_T_REAL	Value of the jerk (always positive) used to halt slave axis after disengage from master axis
VAR_OUTPUT			
B	Done	MC_T_BOOL	Disengaging completed
E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	MC_T_WORD	Error identification



Return

MC_NO_ERROR if successful.
MC_ERR_INVALID_PLCOPEN_STATE
MC_ERR_DEC_OUT_OF_RANGE

Comments

- Divergent to the PLCopen standard the slave axis will be stopped after disengaging from master axis. The parameters ´Deceleration´ and ´Jerk´ are used to calculate the break ramp.
- After disengage the PLCOpenState of the slave axis is set to *MC_PLCOPEN_STATE_DISC_MOTION* and after the axis is stopped to *MC_PLCOPEN_STATE_STAND_STILL.*
- The PLCOpenState of the master axis isn´t changed.

Example

```
MC_CAM_OUT_T    CamOut;

CamOut.Axis = pAxis;
CamOut.Axis = &pSlaveAxis->pFb->Axis;
CamOut.Deceleration = 500;

CamOut.Execute = MC_TRUE;
CamOut.OnCycle();
```

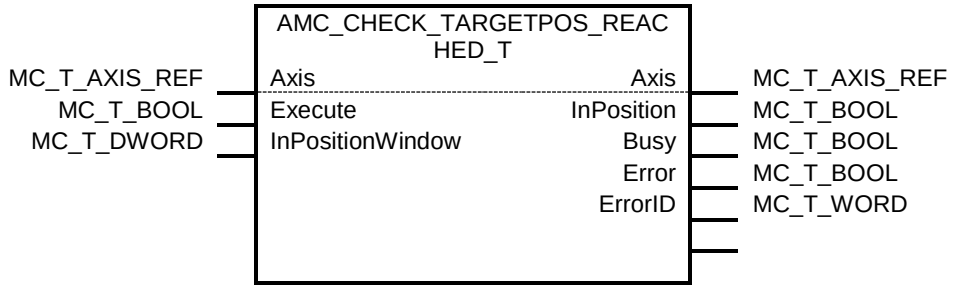
Important definitions

```
class _MC_API MC_CAM_OUT_T
```

4.4 Extension functions

4.4.1 AMC_CHECK_TARGETPOS_REACHED_T

Class		AMC_CHECK_TARGETPOS_REACHED_T		
This Function Block will check if the difference between the commanded position and actual position not greater than the provided InPositionWindow value.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Check target position if inside InPositionWindow at rising edge
	B	InPositionWindow	MC_T_DWORD	In-Position window in drive increments
VAR_OUTPUT				
	B	InPosition	MC_T_BOOL	Actual position near Commanded position. Tolerance window: InPositionWindow
	E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
	B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
	E	ErrorID	MC_T_WORD	Error identification
Note:.The function blocks MC_MOVE_ABSOLUTE_T and MC_MOVE_RELATIVE_T are setting the “Done” output already after the whole trajectory is commanded to the axis. Due to the natural following error, “Done” is set before the target position is reached. By using this function block the application can check the actual position against the commanded position.				

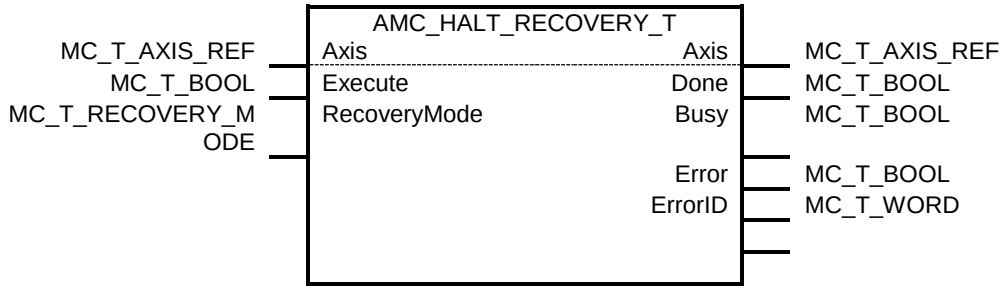


4.4.2 AMC_HALT_RECOVERY_T

Class		AMC_HALT_RECOVERY_T		
This Function Block abort the current movement in case of a) RecoveryMode is equal to MC_RECOVERY_ABORT_MOVEMENT and b) PLCOpenState is equal to MC_PLCOPEN_STATE_DISC_MOTION As a result the target position is set to the actual position (axis doesn´t move anymore) and the PLCOpenState is set to MC_PLCOPEN_STATE_STAND_STILL.				
VAR_IN_OUT				
	B	Axis	MC_T_AXIS_REF	Reference to the axis
VAR_INPUT				
	B	Execute	MC_T_BOOL	Start recovery
	B	RecoveryMode	MC_T_RECOVERY_MODE	Recovery mode: MC_RECOVERY_NO_ACTION or MC_RECOVERY_ABORT_MOVEMENT
VAR_OUTPUT				
	B	Done	MC_T_BOOL	Recovery done

E	Busy	MC_T_BOOL	The FB is not finished and new output values are to be expected
B	Error	MC_T_BOOL	Signals that an error has occurred within the Function Block
E	ErrorID	MC_T_WORD	Error identification

Note:.The function blocks MC_MOVE_ABSOLUTE_T and MC_MOVE_RELATIVE_T are setting the "Done" output already after the whole trajectory is commanded to the axis. Due to the natural following error, "Done" is set before the target position is reached. By using this function block the application can check the actual position against the commanded position.



4.4.3 MC_CalcMoveProfile

Calculate move times and segment distances for a specific movement without moving the axis. To evaluate the time at a certain position use MC_CalcMoveTimeAtPos.

.

```
MC_T_WORD MC_CalcMoveProfile(
    MC_T_AXIS_REF*    pAxis,
    MC_T_REAL         fDistance,
    MC_T_REAL         fVelocity,
    MC_T_REAL         fAcceleration,
    MC_T_REAL         fDeceleration,
    MC_T_REAL         fJerk,
    MC_T_MOVEMENT*    pMove
);
```

Parameters

pAxis
[in] Pointer to axis reference
fDistance
[in] Distance in mm
fVelocity
[in] Maximum velocity in mm/s
fAcceleration
[in] Maximum acceleration in mm/s^2
fDeceleration
[in] Maximum deceleration in mm/s^2
fJerk
[in] Maximum jerk in mm/s^3
pMove
[out] Data structure for movement. Required for MC_CalcMoveTimeAtPos

Return

MC_NO_ERROR if successful.

Example

```
MC_T_MOVEMENT   oMove;
pAxis->dwCycleTime = 1000;
```

```
pAxis->dwIncPerMM =  10000;
MC_CalcMoveProfile(pAxis, 100, 250, 1000, 1000, 0, &oMove);
```

4.4.4 MC_CalcMoveTimeAtPos

Calculate time until a certain position is reached. MC_CalcMoveProfile has to be called prior calling this function.

MC_T_WORD MC_CalcMoveTimeAtPos(
MC_T_AXIS_REF* pAxis,
MC_T_MOVEMENT* pMove,
MC_T_REAL fTriggerDist,
MC_T_INT64* pTriggerTime
);

Parameters

pAxis
[in] Pointer to axis reference
pMove
[in+out] Data structure for movement. Required for MC_CalcMoveTimeAtPos
fTriggerDist
[in] Trigger distance based on start of movement
wNewReqDevState
[in] Requested state
pTriggerTime
[out] Trigger time in milliseconds

Return

MC_NO_ERROR if successful.

Comment

Data provided via pMove will be modified by this function!
For consecutively usage of this function the *fTriggerDist* requires increasing values like shown in the example.

Example

```
MC_T_INT64      ulTriggerTime;
MC_CalcMoveTimeAtPos(pAxis, &oMove, 15, &ulTriggerTime);
MC_CalcMoveTimeAtPos(pAxis, &oMove, 25, &ulTriggerTime);
MC_CalcMoveTimeAtPos(pAxis, &oMove, 55, &ulTriggerTime);
```

4.4.5 MC_DriveSetTargetStep

Set velocity without using build-in trajectory generator.

EC_T_DWORD MC_DriveSetTargetStep(
MC_T_AXIS_REF* pAxis,
MC_T_POSITION lTargetPosStep,
MC_T_VELACC lTargetVel,
MC_T_VELACC lTargetAcc
);

Parameters

pAxis
[in] Pointer to axis reference
lTargetStep
[in] Increments (positive or negative) added to current target position

lTargetVel
[in] Optional: Final velocity for feed forward
lTargetAcc
[in] Optional: Final acceleration for feed forward

Return
MC_NO_ERROR if successful.
MC_ERR_INVALID_OPERATION_MODE

Comment
Don´t use this function in parallel to MoveAbsolute, MoveRelative, MoveVelocity, etc.
Function should be called just once in a network cycle.
Application has to execute MC_POWER_T as well.

4.4.6 ELMO extension functions

4.4.6.1 SetGainScheduling

Sets the gain scheduling manual index for given slave (drive controller). The gain scheduling is stored in the object 0x2E00.

```
MC_T_DWORD SetGainScheduling(
    MC_T_WORD       wStationAddress,
    MC_T_WORD       wValue
);
```

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_WORD	wValue,	Gain value

Return

MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

Comment

This function may not be called in tEcJobTask!

4.4.6.2 SetSmoothFactor

Sets the smooth factor for given slave (drive controller). The smooth factor is stored in the object 0x31D9:1.

```
MC_T_DWORD SetSmoothFactor(
    MC_T_WORD       wStationAddress,
    MC_T_DWORD      dwValue
);
```

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_DWORD	dwValue,	Smooth factor value

Return

MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

Comment

This function may not be called in tEcJobTask!

4.4.6.3 SetUserInteger

Stores an integer value into internal array for given slave (drive controller). There are up to 24 slots to store. To read a stored value the GetUserInteger can be used.

```
MC_T_DWORD SetUserInteger(
    MC_T_WORD       wStationAddress,
    MC_T_BYTE       bySubIndex,
    MC_T_DWORD      dwValue
);
```

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_BYTE	bySubIndex,	Subindex 1... 24
MC_T_DWORD	dwValue,	Integer value to store

Return

MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

Comment
This function may not be called in tEcJobTask!

4.4.6.4 GetUserInteger

Reads an integer value from internal array for given slave (drive controller). There are up to 24 slots to store. To store an integer value the SetUserInteger can be used.

```
MC_T_DWORD GetUserInteger(
    MC_T_WORD       wStationAddress,
    MC_T_BYTE       bySubIndex,
    MC_T_DWORD*     pdwValue
);
```

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_BYTE	bySubIndex,	Subindex 1... 24
MC_T_DWORD*	pdwValue,	Pointer to buffer

NOTE: This function may not be called in tEcJobTask!

Return
MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

Comment
This function may not be called in tEcJobTask! The memory for the buffer has to be allocated prior this function call.

4.4.6.5 SetUserFloat

Stores a float value into internal array for given slave (drive controller). There are up to 24 slots to store. To read a stored value the GetUserFloat can be used.

```
MC_T_DWORD SetUserFloat(
    MC_T_WORD       wStationAddress,
    MC_T_BYTE       bySubIndex,
    MC_T_REAL       fValue
);
```

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_BYTE	bySubIndex,	Subindex 1... 24
MC_T_REAL	fValue,	Float value to store

Return
MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

Comment
This function may not be called in tEcJobTask!

4.4.6.6 GetUserFloat

Reads a float value from internal array for given slave (drive controller). There are up to 24 slots to store. To store an integer value the SetUserFloat can be used.

MC_T_DWORD GetUserFloat(
MC_T_WORD wStationAddress,
MC_T_BYTE bySubIndex,
MC_T_REAL* pfValue
);

Parameters

MC_T_WORD	wStationAddress,	EtherCAT station address
MC_T_BYTE	bySubIndex,	Subindex 1... 24
MC_T_REAL*	pfValue,	Pointer to buffer

Return

MC_T_DWORD, error code in case of error, EC_E_NOERROR otherwise

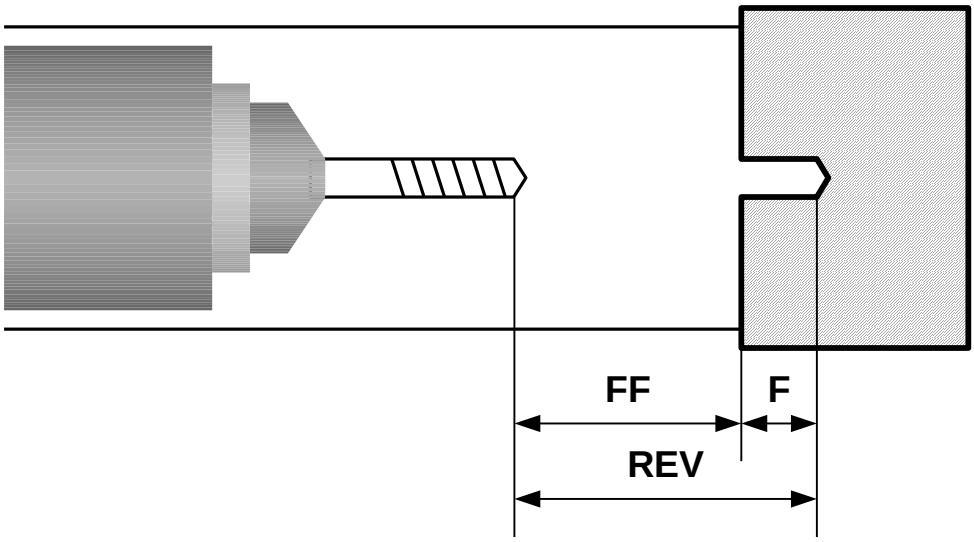
Comment

This function may not be called in tEcJobTask! The memory for the buffer has to be allocated prior this function call.

Bibliography

[1] PLCopen, „Function blocks for motion control, version 2.0,“ [Online]. Available: www.plcopen.org.
[2] acontis technologies GmbH, „EC-Master - EtherCAT Master Stack Class B - manual,“ 2014.
[3] acontis technologies GmbH, EC-Motion quick start guide, 2014.

Appendix A



This is a **simple example of drilling a hole**. We use a single axis (MC_T_AXIS_REF) and 3 daisy chained MCFB's:

FB1: MC_MOVE_ABSOLUTE_T
FB2: MC_MOVE_RELATIVE_T
FB3: MC_MOVE_ABSOLUTE_T

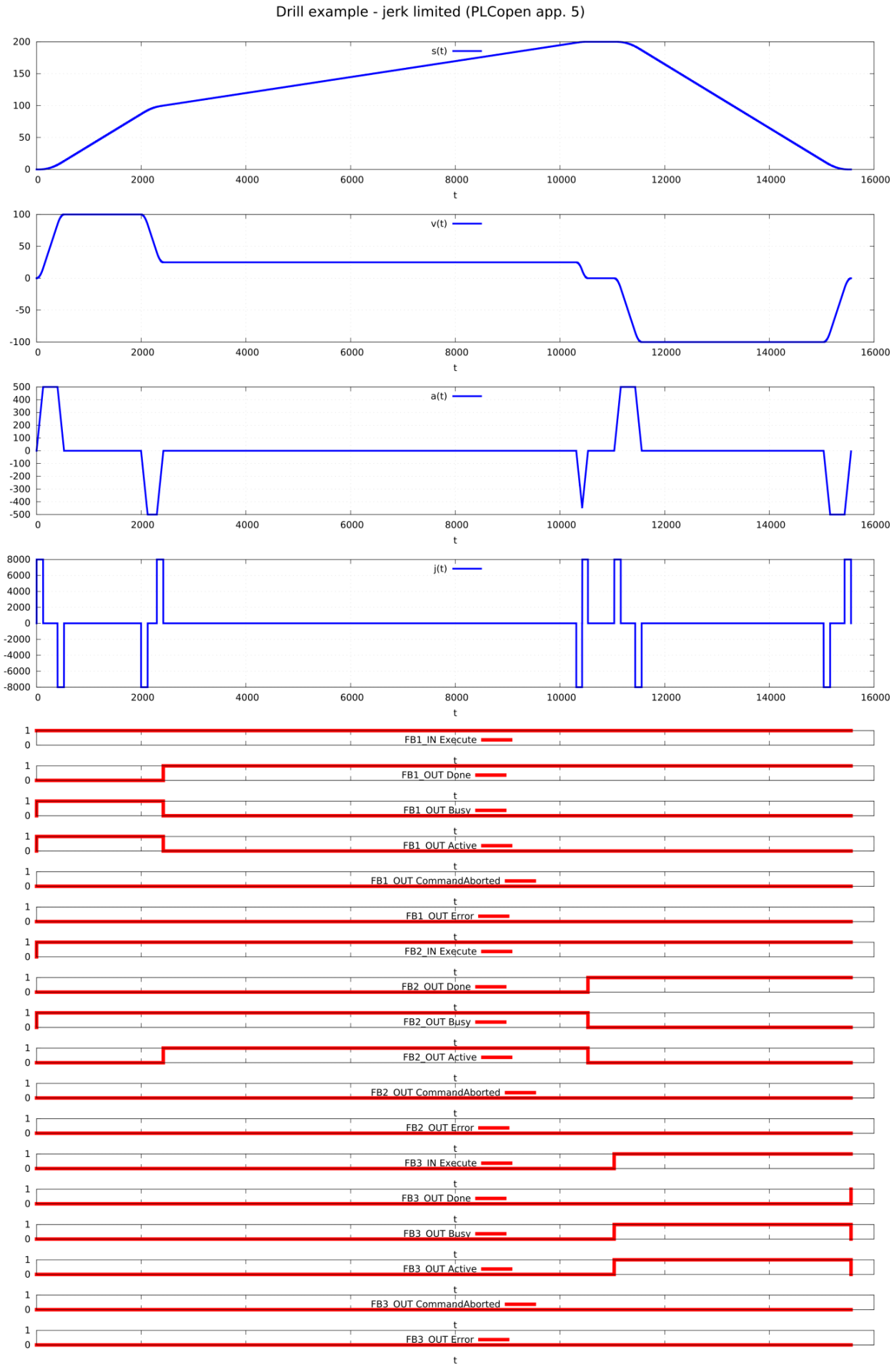
In order to drill the hole, the following steps have to be done:

Step 1: Initialization, for instance at power up.
Step 2: Move forward to drilling position and start the drill turning. In this way it will be fully operational before the position is reached and then check if both actions are completed.
Step 3: Drill the hole.
Step 4: After drilling the hole we have to wait for the step-chain sequence to finish dwelling to free the hole of any debris, which might have been stuck in the hole.
Step 5: Move drill back to starting position and shut the spindle off. Combining the completion of moving backwards and stopping the spindle we signal the step-chain to start over.

The motion parameters for the MCFB's are parameterized as follows:

	Position/ Distance	**Velocity**	**Acceleration/ Deceleration**	**Jerk**	**BufferMode**
FB1 (MC_MOVE_ABSOLUTE_T)	100	100	500	8000	MC_ABORTING
FB2 (MC_MOVE_RELATIVE_T)	100	25	500	8000	MC_BLENDING_LOW
FB3 (MC_MOVE_ABSOLUTE_T)	0	100	500	8000	MC_ABORTING

The following graph shows the generated trajectory (first Position, 2nd Velocity, 3rd Acceleration, 4th Jerk and digital outputs), calculated by the EC-Motion kernel.


Drill example - jerk limited (PLCopen app. 5)
s(t)
v(t)
a(t)
j(t)
t
FB1_IN Execute
FB1_OUT Done
FB1_OUT Busy
FB1_OUT Active
FB1_OUT CommandAborted
FB1_OUT Error
FB2_IN Execute
FB2_OUT Done
FB2_OUT Busy
FB2_OUT Active
FB2_OUT CommandAborted
FB2_OUT Error
FB3_IN Execute
FB3_OUT Done
FB3_OUT Busy
FB3_OUT Active
FB3_OUT CommandAborted
FB3_OUT Error