Motion Control Library in “C++”

Controlling Drives based on CiA 402 or SERCOS device profile

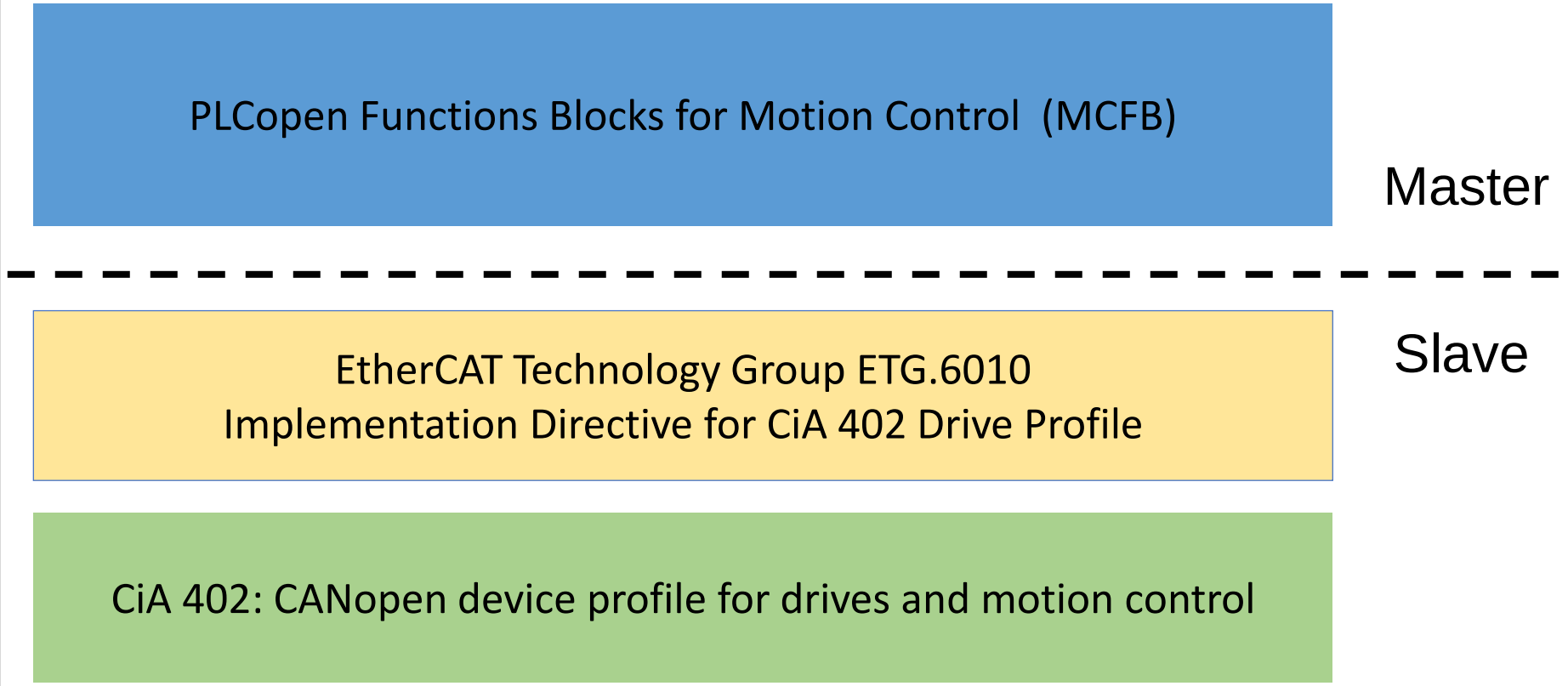
Content

- Introduction to PLCopen and CiA 402
- EC-Motion Library Architecture
- Administrative Functions Blocks
- Single Axis Motion Functions Blocks
- Camming Function Blocks
- Examples
- Highlights

Introduction

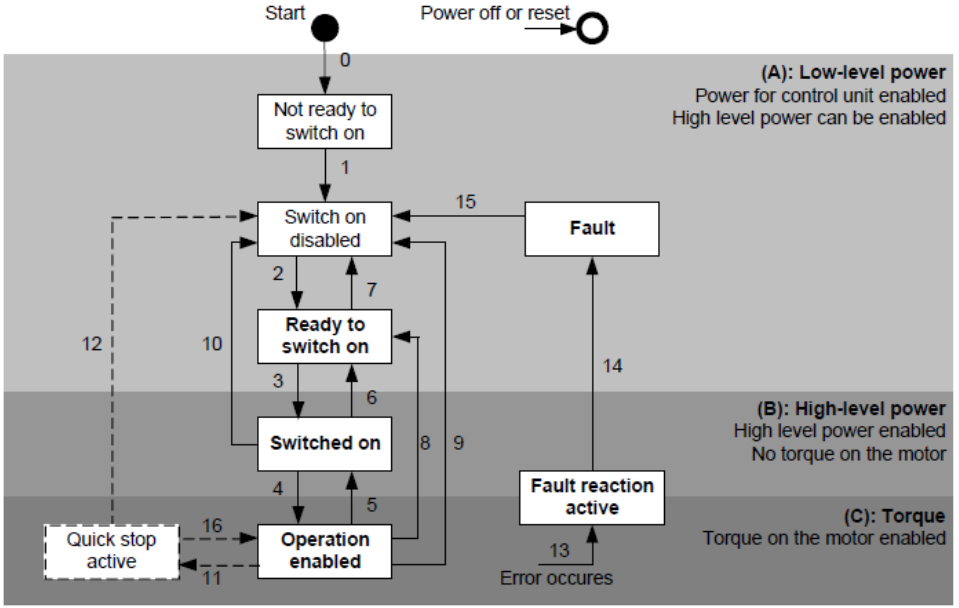
- Most available drives with EtherCAT slave interface are based on the CiA 402 standard, e. g., Yaskawa, Copley, Omron, …
- CiA 402 organizes parameters in a so called object dictionary and a drive state machine
- Based on this definitions it shall be possible to run drives from different manufacturers with the same application
- EtherCAT Technology Group (ETG) document ETG6010_V1i0i0_D_R_CiA402_ImplDirective gives additional implementation hints for using CiA 402 with EtherCAT
- The organization PLCopen has defined Functions Blocks for Motion Control to simplify usage of motion functions within a PLC program

Introduction Standards



CiA 402: CANopen device profile for drives and motion control

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



ETG Implementation Directive for CiA 402 Drive Profile

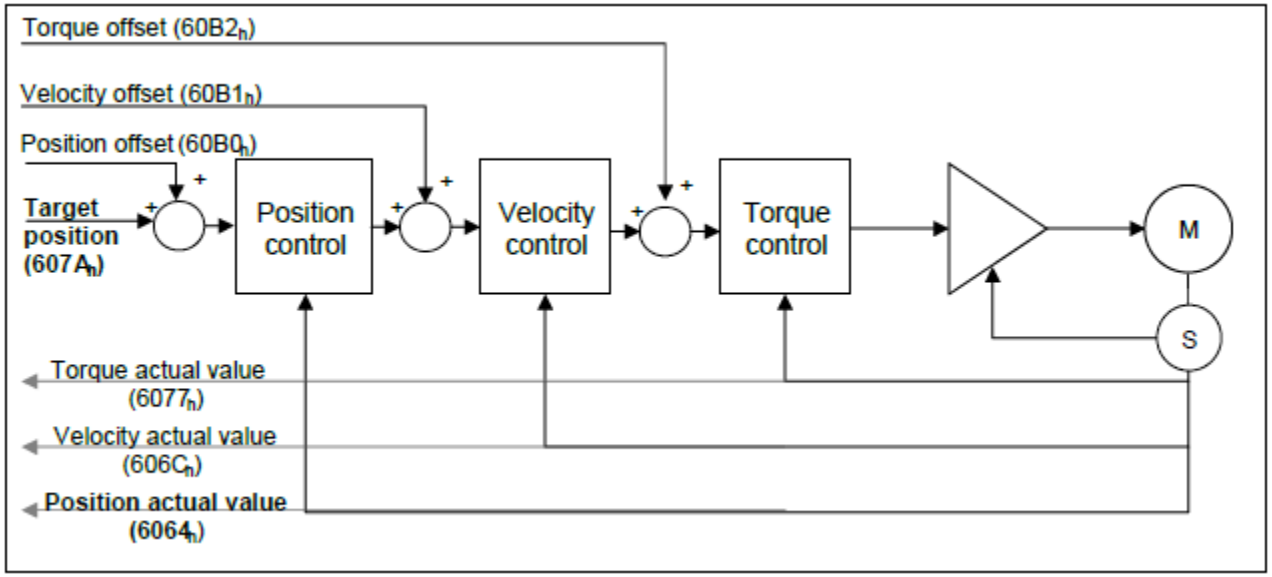
- The drive has to support at least on of the cyclic operation modes:
  CSP or CSV or CST

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

ETG Implementation Directive for CiA 402 Drive Profile

CSP: Cyclic Synchronous Position Mode

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



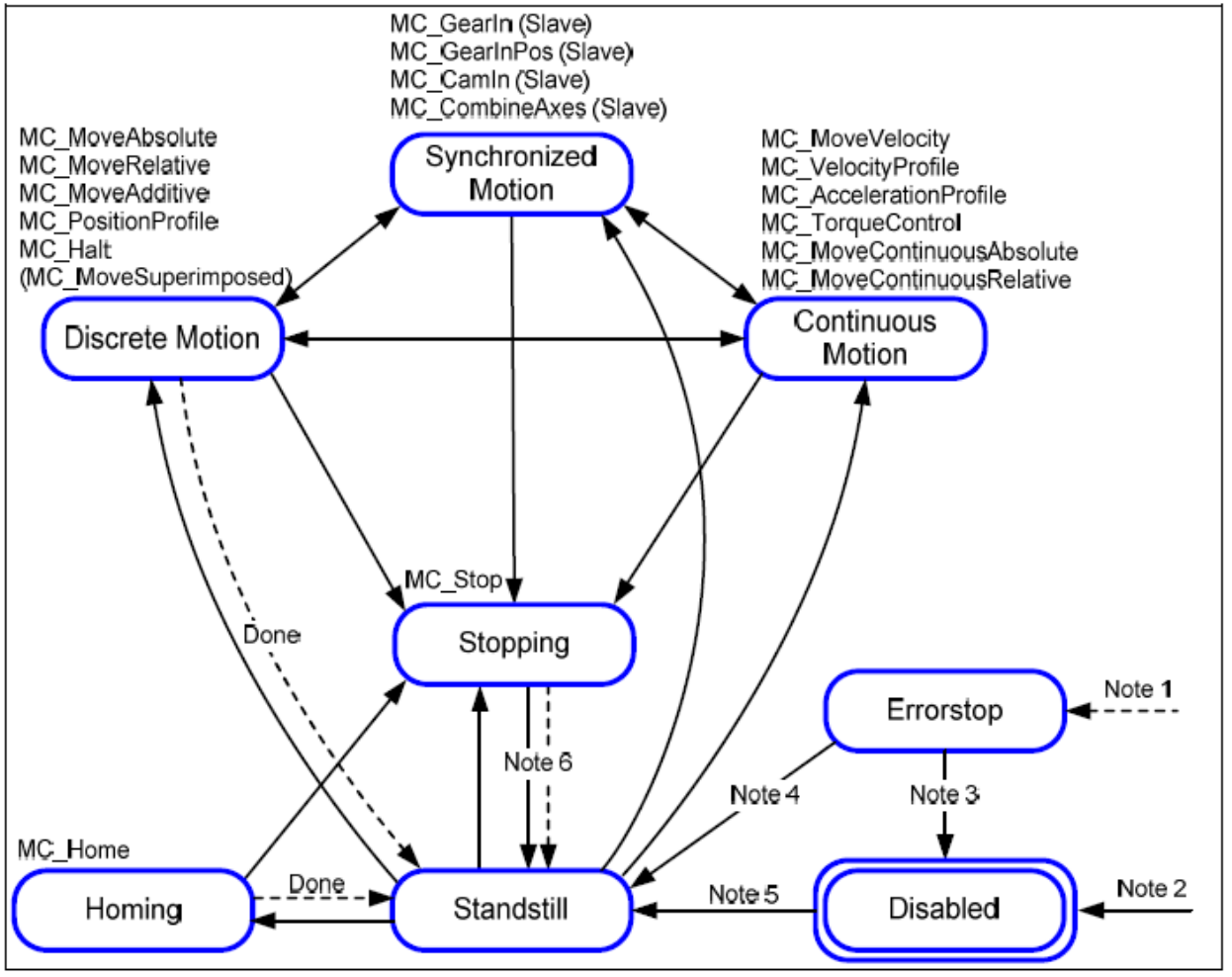
PLCopen Functions Blocks for Motion Control

- **PLCopen**, as an organization active in Industrial Control, is creating a higher efficiency in your application software development and lowering your life-cycle costs. As such it is based on standard available tools to which extensions are and will be defined. With results like Motion Control Library, Safety, XML specification, Reusability Level and Conformity Level, PLCopen made solid contributions to the community, extending the hardware independence from the software code, as well as reusability of the code and coupling to external software tools.
  http://www.plcopen.org/
- **PLCopen Motion Control Specifications**
  PLCopen motion standard provide a way to have standard application libraries that are reusable for multiple hardware platforms.

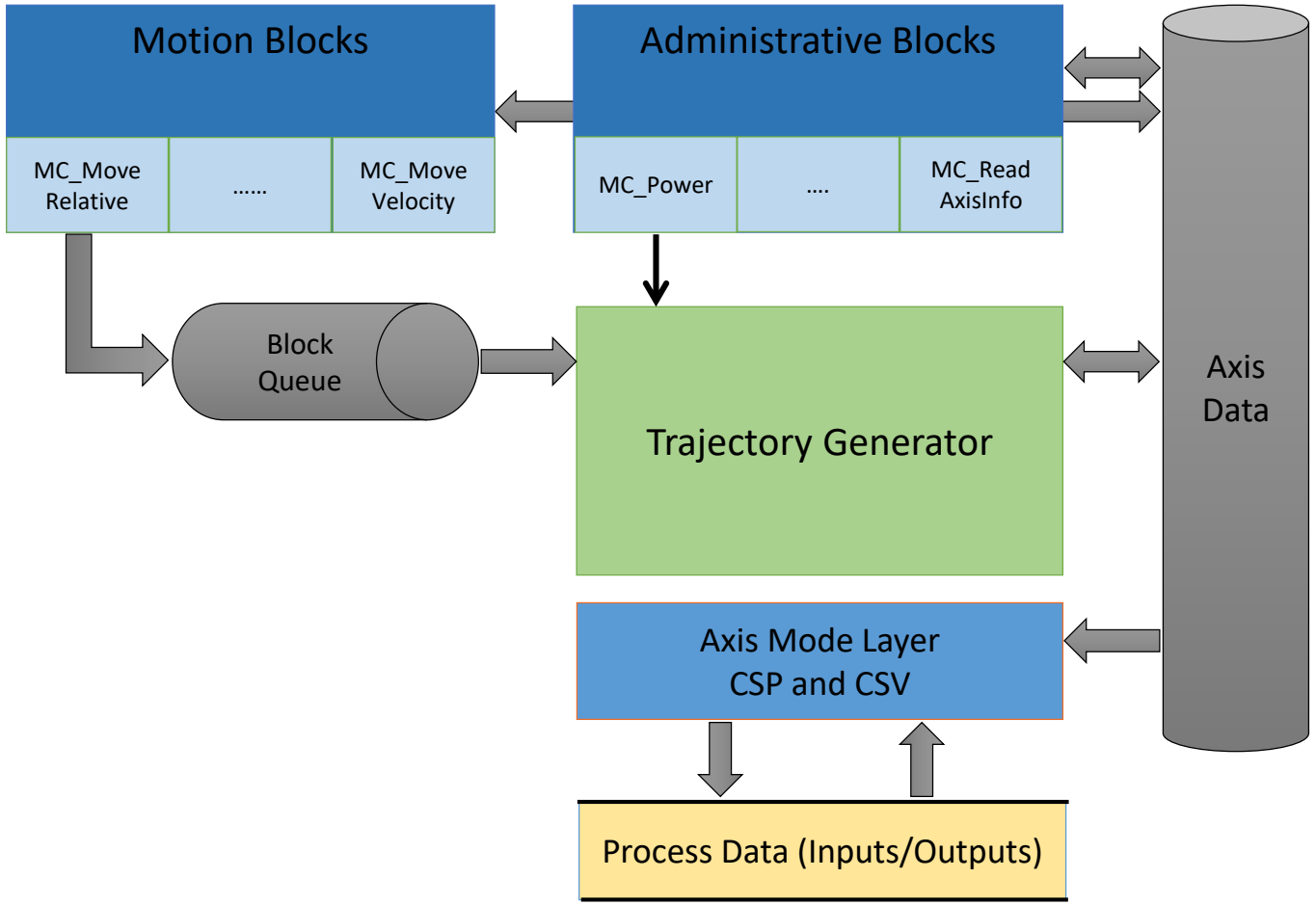
  Part 1 and 2: Function blocks for motion control V2.0
  http://www.plcopen.org/pages/tc2_motion_control/

Introduction
PLCopen: Motion state machine

PLCopen Functions Blocks for Motion Control



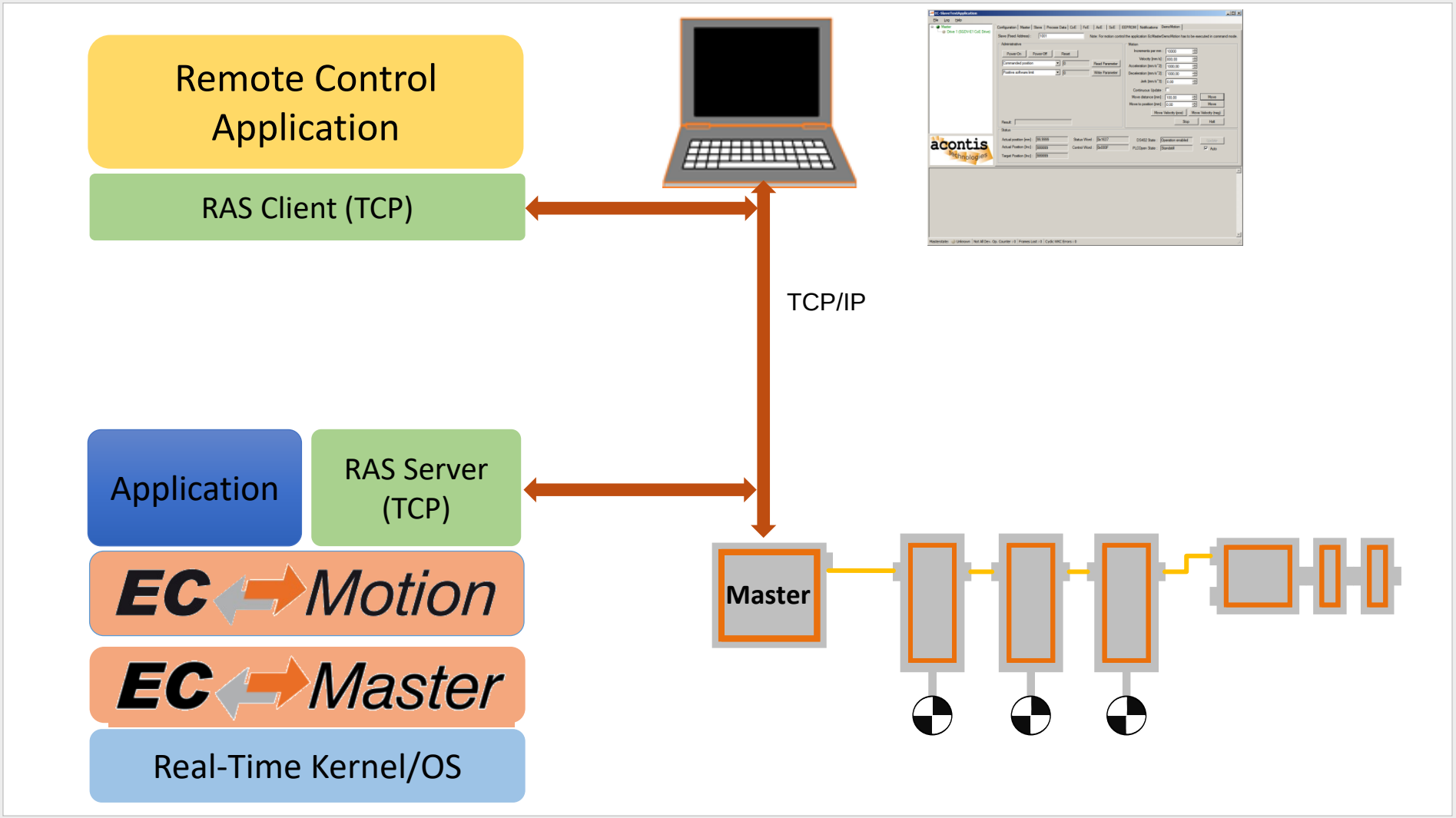
EC-Motion Control Library Architecture



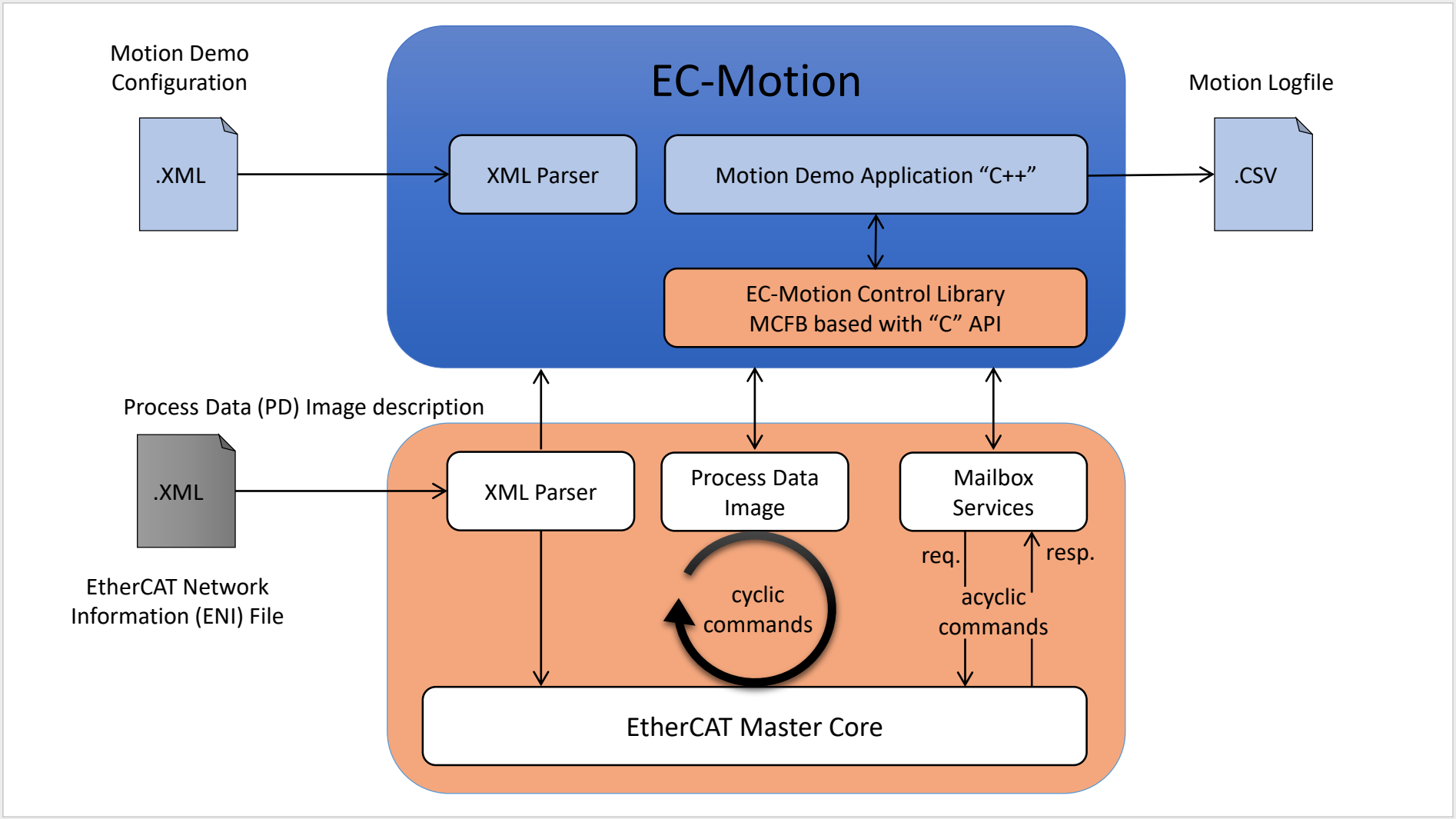
EC-Motion Product Overview

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

EC-Motion System Architecture



Software Architecture



Administrative Functions Blocks (1)

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

Administrative Functions Blocks (2)

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

Single Axis Motion Functions Blocks (1)

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

Single Axis Motion Functions Blocks (2)

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

Camming Functions Blocks

- With Multi-Axis Function Blocks a synchronized relationship exists between two or more axes. The synchronization can be related to time or position. Often this relationship is between a master axis and one or more slave axes. A master axis can be a virtual axis.
- From the state diagram point of view, the multi-axis Function Blocks related to Camming can be looked at as a master axis in one state (for instance: MC_MOVEVELOCITY_T) and the slave axis in a specific synchronized state, called 'SychronizedMotion'.

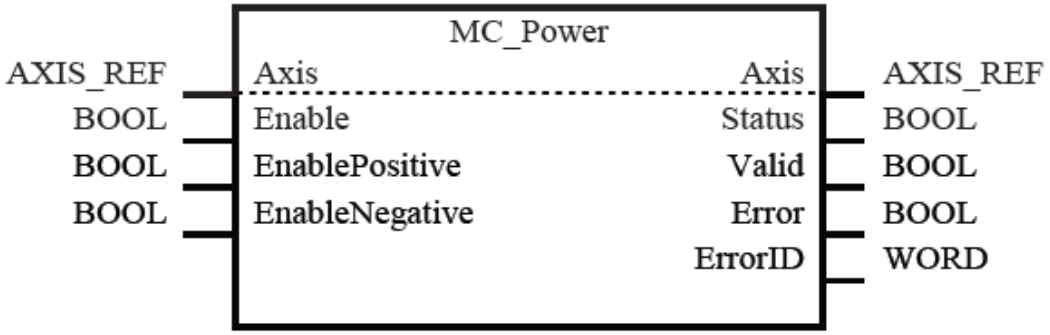
- **MC_CAMTABLE_SELECT_T**:
  Selects the CAM tables by setting the connections to the relevant tables.

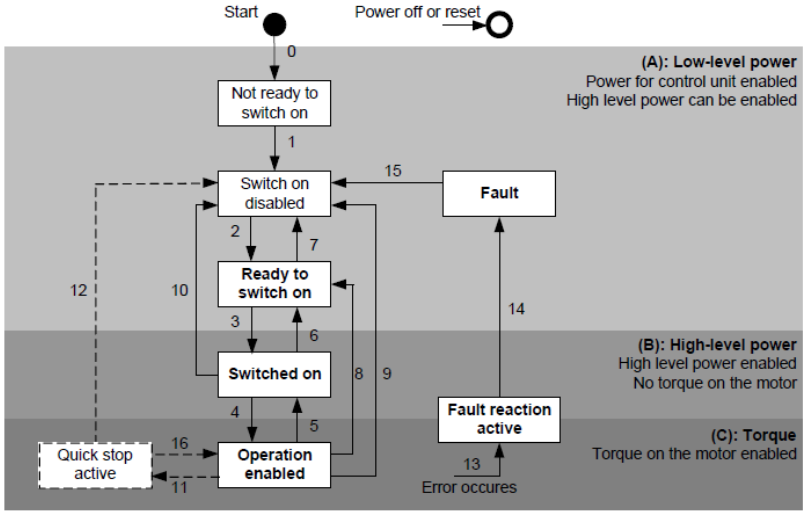
- **MC_CAM_IN_T**:
  Engages the CAM.

- **MC_CAM_OUT_T** :
  This Function Block disengages the Slave axis from the Master axis immediately.
  The Slave axis will be stopped if moving.

Example: MC_Power in PLCopen

This Function Block controls the power stage and implements the CiA 402 drive state machine.



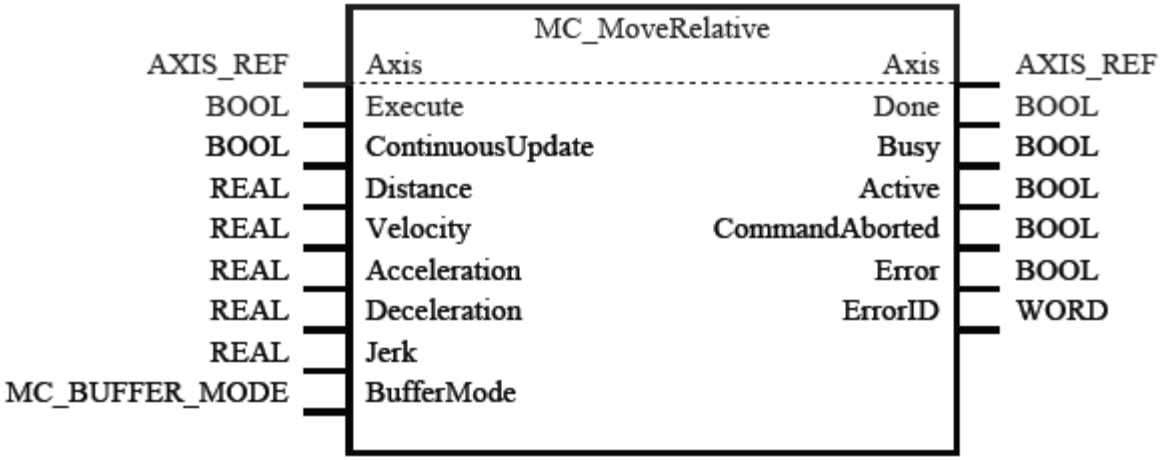


Example: MC_Power in “C++” language

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

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

    void _MC_THIS_API OnCycle();
} _MC_PACKED;


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

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

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

Example: MC_MoveRelative in PLCopen

This Function Block commands a controlled motion of a specified distance relative to the set position at the time of the execution



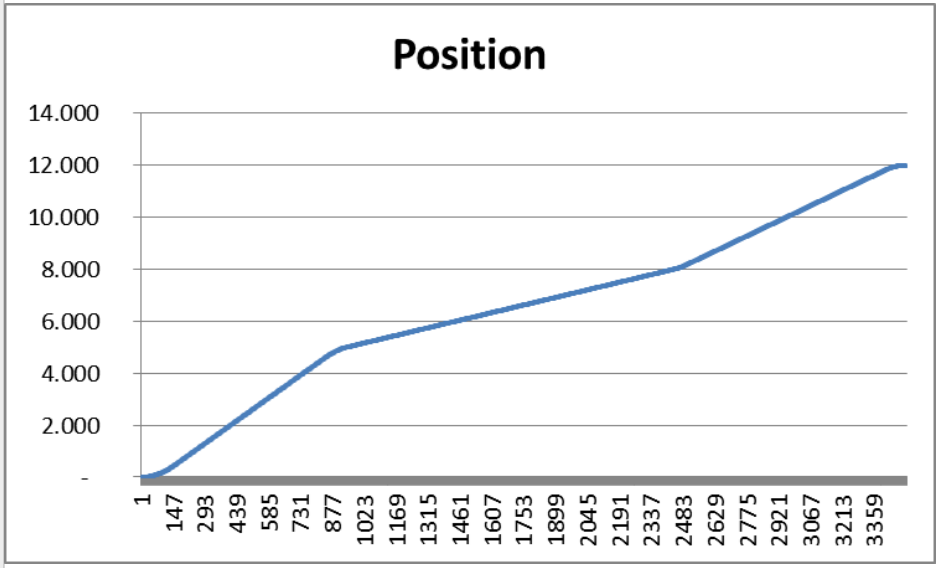
Example: MC_MoveRelative in “C++” language

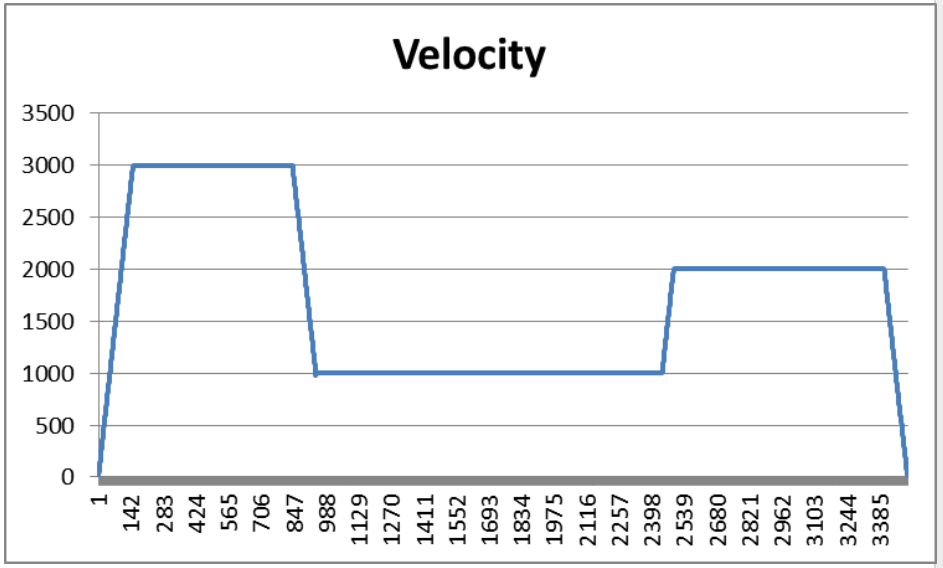
```
class _MC_API MC_MOVE_RELATIVE_T : public MC_BUFFERED_FB_T
{
public:
    // OUT's
    const MC_T_BOOL     &Done;                  /* OUT(B): The axis is within a range close to the target position */
    const MC_T_BOOL     &Busy;                  /* OUT(E): The FB is not finished and new output values are to be expected */

    // IN's
    MC_T_BOOL           Execute;                /* IN(B): Start the motion at rising edge */
    MC_T_BOOL           ContinuousUpdate;       /* IN(E): Continuous Update (Trapezoid profile only) */
    MC_T_REAL           Distance;               /* IN(B): Relative distance for the motion */
    MC_T_REAL           Velocity;               /* IN(E): Value of the max velocity (always positive, not necessarily reached). */
    MC_T_REAL           Acceleration;           /* IN(E): Value of the acc (always positive, increasing energy of the motor). */
    MC_T_REAL           Deceleration;           /* IN(E): Value of the dec (always positive, decreasing energy of the motor). */
    MC_T_REAL           Jerk;                   /* IN(E): Value of the Jerk (always positive). */

    MC_MOVE_RELATIVE_T(MC_T_AXIS_REF *pAxis = MC_NULL);

    void _MC_THIS_API OnCycle();
}
```

Example MC_MoveRelative Buffermode = MC_BLENDING_LOW

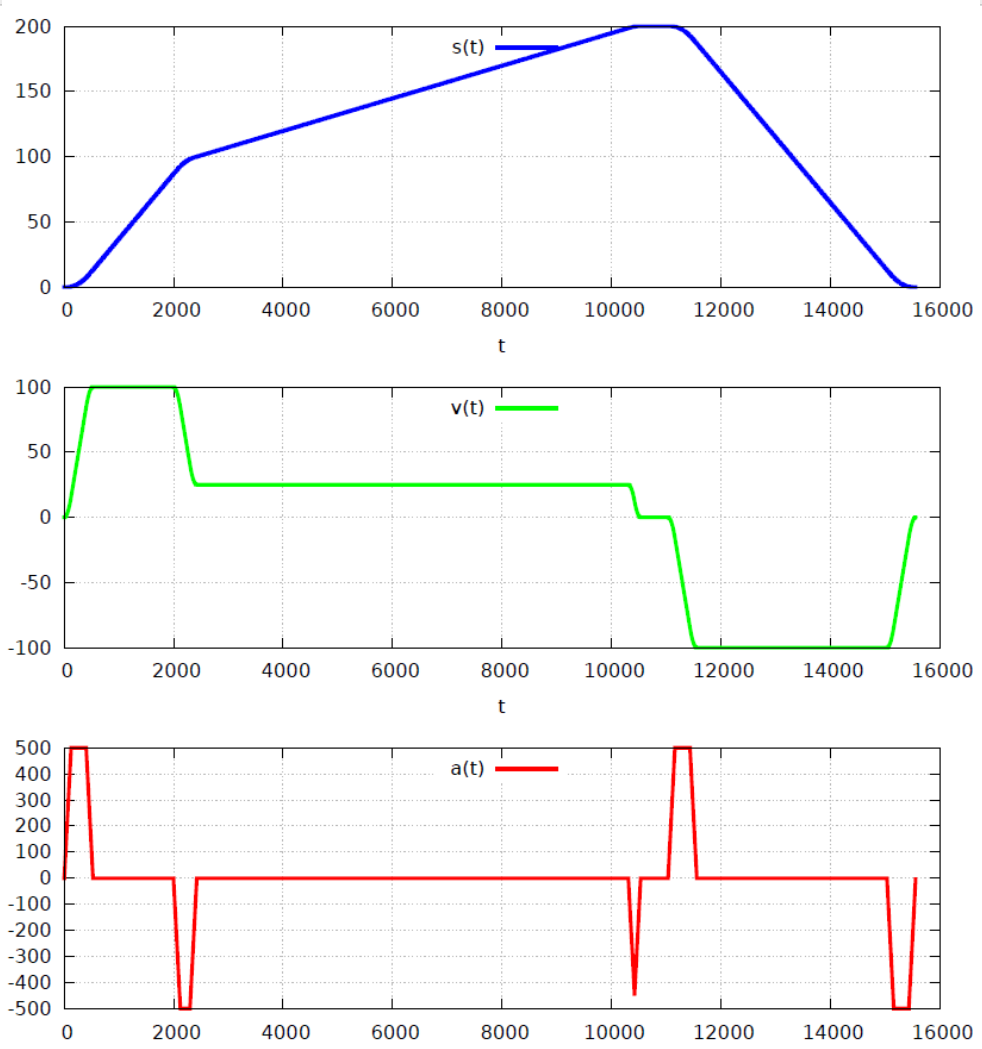
Sequence of three MC_MoveRelative without stop

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





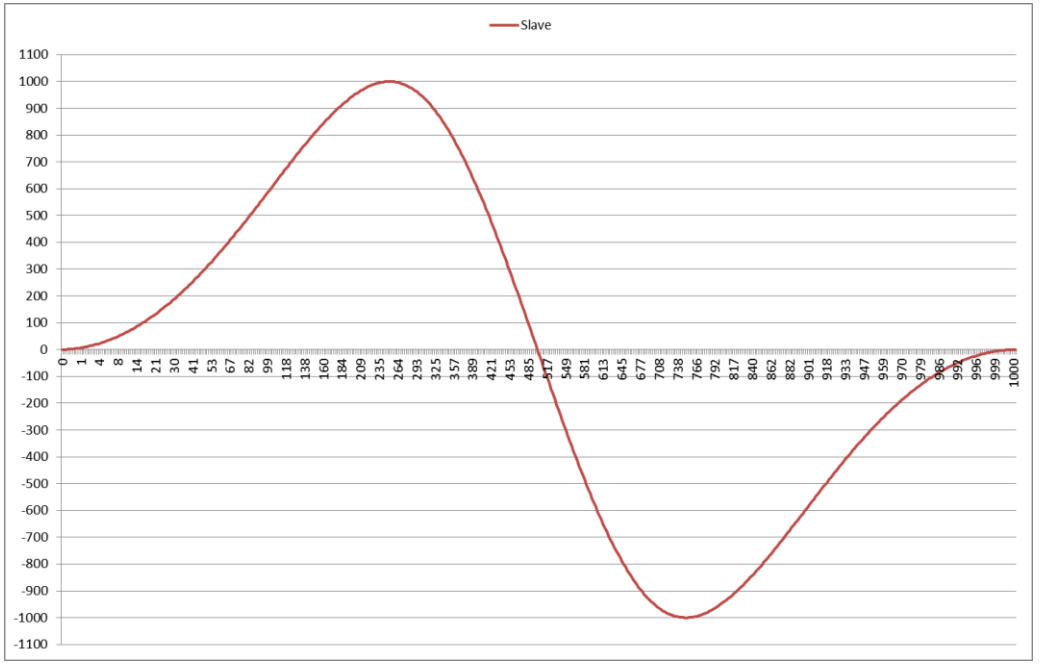
Drill example – jerk limited



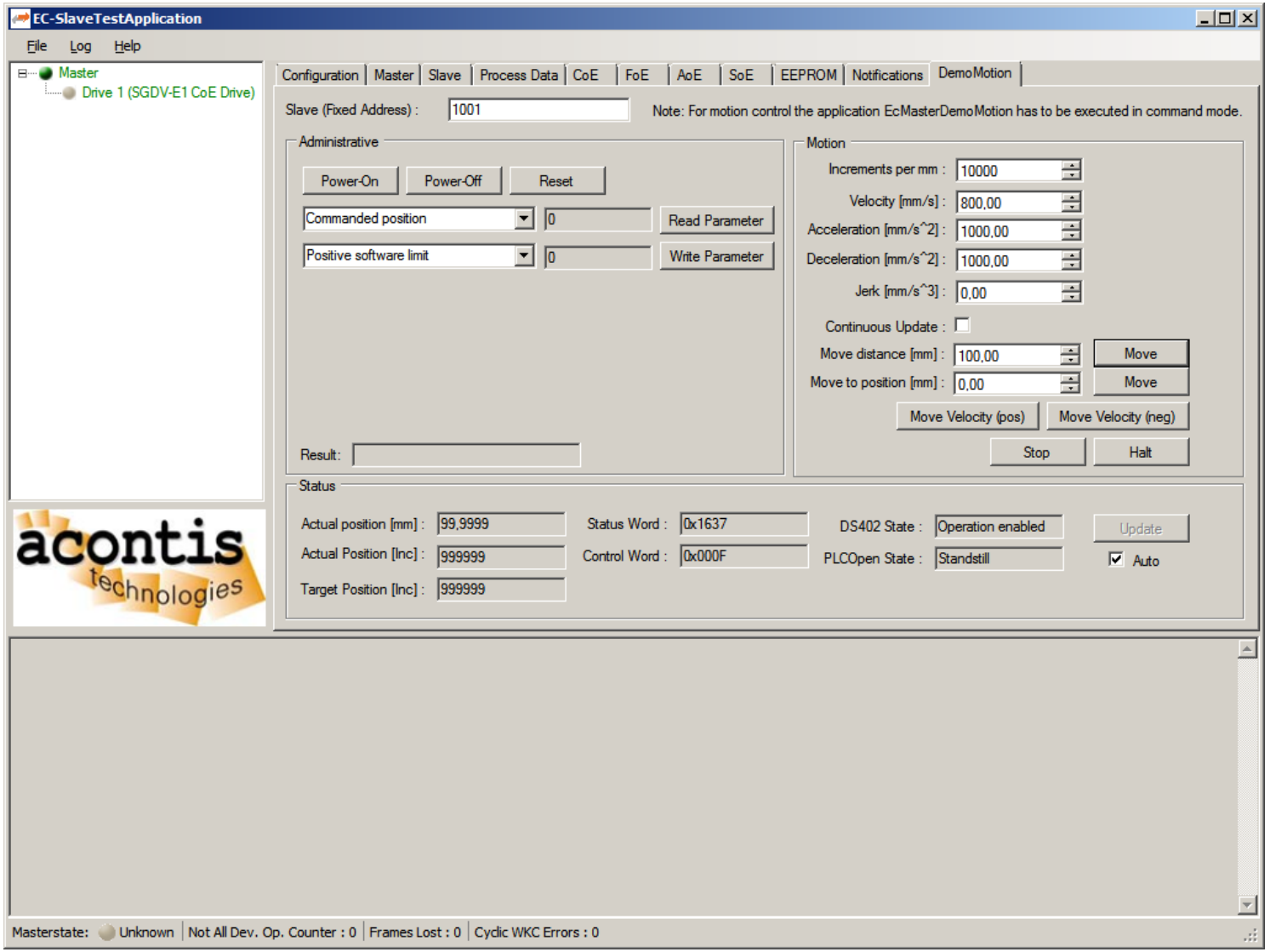
Camming Example

EC Master

```
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;
```



Remote Control with EC-STA
DemoMotion tab to command functions



Highlights

- CiA402 and SERCOS Profile
- Independent from communication layer (EtherCAT, CAN)
- Jerk limited movements
- Changing parameters during movement (continuous update)
- Software limits
- Buffer modes (buffered, blending)
- Operating modes
  - Cyclic Synchronous Position (CSP)
  - Cyclic Synchronous Velocity (CSV)
  - Profile Position (PP)
- Virtual axis
- Efficient implementation → Low CPU load
- According to PLCopen Standard V2.0
- Library includes source code