Manual

TwinCAT MC Flying Saw

TwinCAT 3

Version: 1.1
Date: 2016-06-27
Order No.: TF5055

BECKHOFF

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with the applicable national standards.
It is essential that the following notes and explanations are followed when installing and commissioning these components.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics.
In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE®, XFC®and XTS® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

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

Copyright

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

 **DANGER**	**Serious risk of injury!** Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.
 **WARNING**	**Risk of injury!** Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.
 **CAUTION**	**Personal injuries!** Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.
 **Attention**	**Damage to the environment or devices** Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.
 **Note**	**Tip or pointer** This symbol indicates information that contributes to better understanding.

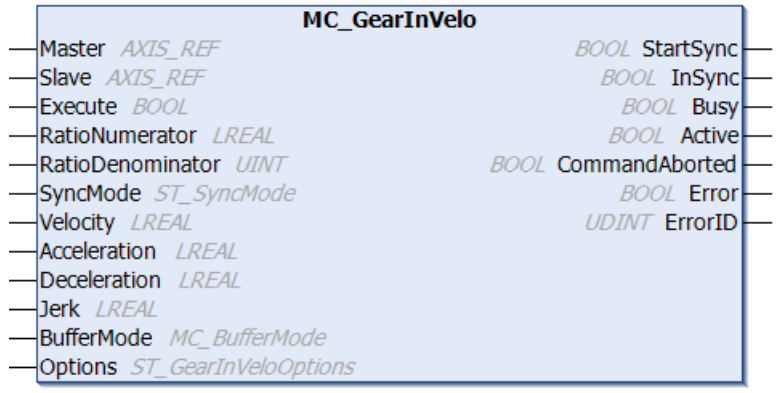
2 Overview

In many plants workpieces undergo machining operations while being transported. For this purpose it is necessary to synchronise the position and the speed of tool and workpiece, so that the tool can then be applied as if to a stationary workpiece. One example of such an application is a saw that during the transport process cuts through the material that is being transported (flying saw). In order to implement this kind of application, TwinCAT provides the flying saw.

The TwinCAT PLC library Tc2_MC2_FlyingSaw, available as an *additional product*, provides easy management of the flying saw. An example program using the flying saw [▶ 14] makes use of this library.

3 Flying saw

3.1 MC_GearInVelo



The function block *MC_GearInVelo* activates a linear master-slave coupling (gear coupling). If the master axis is already moving, the slave axis synchronizes to the master velocity. The block accepts a fixed gear ratio in numerator/denominator format.

The slave axis can be uncoupled with the function block MC_GearOut. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC_Stop or MC_Halt.

Inputs

```
VAR_INPUT
    Execute          : BOOL;
    RatioNumerator   : LREAL;
    RatioDenominator : UINT;
    SyncMode         : ST_SyncMode;
    Velocity         : LREAL;
    Acceleration     : LREAL;
    Deceleration     : LREAL;
    Jerk             : LREAL;
    BufferMode       : MC_BufferMode;
    Options          : ST_GearInVeloOptions;
END_VAR
```

Execute:The command is executed with a rising edge at input *Execute*.

RatioNumerator: Gear ratio numerator. Alternatively, the gear ratio can be specified in the enumerator as a floating comma value, if the denominator is 1.

RatioDenominator: Gear ratio denominator

SyncMode: In the data structure SyncMode [▶ 12] boundary conditions for the synchronization process are specified via individual flags.

Velocity: Maximum slave velocity in the synchronization phase. If a velocity is not specified, the maximum velocity of the axis from the system manager data is used. **Notice:** The velocity given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

Acceleration: Maximum slave acceleration in the synchronization phase. If an acceleration is not specified, the maximum acceleration of the axis from the system manager data is used. **Notice:** The acceleration given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

Deceleration: Maximum slave deceleration in the synchronization phase. If a deceleration is not specified, the maximum deceleration of the axis from the system manager data is used. **Notice:** The deceleration given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

Jerk: Maximum slave jerk in the synchronization phase. If a jerk is not specified, the maximum jerk of the axis from the system manager data is used. **Notice:** The jerk given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

BufferMode: Currently not implemented

Options: Currently not implemented

For a 1:4 ratio the *RatioNumerator* must be 1, the *RatioDenominator* must be 4. Alternatively, the *RatioDenominator* may be 1, and the gear ratio can be specified as floating-point number 0.25 under *RatioNumerator*. The *RatioNumerator* may be negative.

Outputs

```
VAR_OUTPUT
    StartSync      : BOOL;
    InSync         : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

StartSync: Becomes TRUE when the synchronization with the master axis was started.

InSync: Becomes TRUE, if the coupling was successfully completed and the slave axis is synchronized with the master axis.

Busy: The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. If *Busy* becomes FALSE again, the function block is ready for a new job. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.

Active: Active indicates that the command is executed (currently Active=Busy, see BufferMode)

CommandAborted: Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number

Inputs/outputs

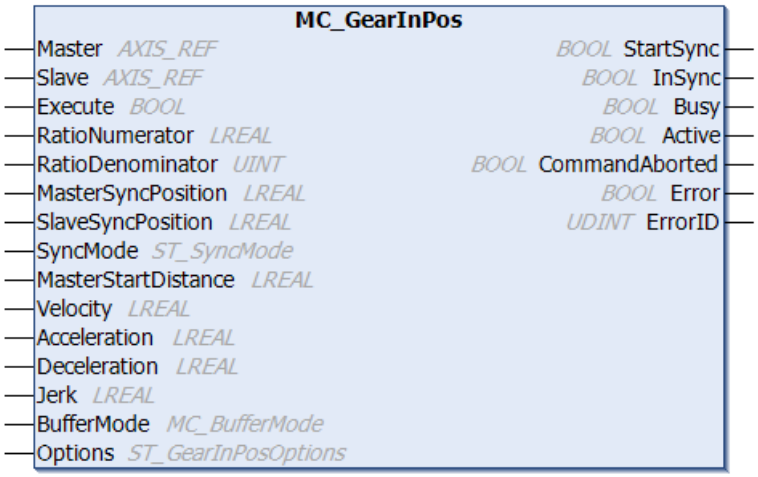
```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Master: Master axis data structure.

Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

3.2 MC_GearInPos



The function block *MC_GearInPos* synchronizes a slave axis precisely with a master axis (flying saw). The synchronization velocity is achieved exactly at the synchronous position of the master and slave.

The master axis must already be moving, otherwise synchronization is not possible.

The slave axis can be uncoupled with the function block MC_GearOut. If the slave is decoupled while it is moving, then it retains its velocity and can be halted using MC_Stop or MC_Halt.

Inputs

```
VAR_INPUT
    Execute             : BOOL;
    RatioNumerator      : LREAL;
    RatioDenominator    : UINT;
    MasterSyncPosition  : LREAL;
    SlaveSyncPosition   : LREAL;
    SyncMode            : ST_SyncMode;
    MasterStartDistance : LREAL;
    Velocity            : LREAL;
    Acceleration        : LREAL;
    Deceleration        : LREAL;
    Jerk                : LREAL;
    BufferMode          : MC_BufferMode;
    Options             : ST_GearInPosOptions;
END_VAR
```

Execute: The command is executed with a rising edge at input *Execute*.

RatioNumerator: Gear ratio numerator. Alternatively, the gear ratio can be specified in the enumerator as a floating comma value, if the denominator is 1.

RatioDenominator: Gear ratio denominator

MasterSyncPosition: The master's synchronous position

SlaveSyncPosition: The slave's synchronous position

SyncMode: In the data structure SyncMode [▶ 12] boundary conditions for the synchronization process are specified via individual flags.

MasterStartDistance: Currently not implemented

Velocity: Maximum slave velocity in the synchronization phase. If a velocity is not specified, the maximum velocity of the axis from the system manager data is used. **Notice:** The velocity given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

Acceleration: Maximum slave acceleration in the synchronization phase. If an acceleration is not specified, the maximum acceleration of the axis from the system manager data is used. **Notice:** The acceleration given here is only checked if this checking is activated through the SyncMode [▶ 12] variable.

Deceleration: Maximum slave deceleration in the synchronization phase. If a deceleration is not specified, the maximum deceleration of the axis from the system manager data is used. **Notice:** The deceleration given here is only checked if the checking is activated through the SyncMode [▶ 12] variable.

Jerk: Maximum slave jerk in the synchronization phase. If a jerk is not specified, the maximum jerk of the axis from the system manager data is used. **Notice:** The jerk given here is only checked if the checking is activated through the SyncMode [▶ 12] variable.

BufferMode: Currently not implemented

Options: Currently not implemented

For a 1:4 ratio the *RatioNumerator* must be 1, the *RatioDenominator* must be 4. Alternatively, the *RatioDenominator* may be 1, and the gear ratio can be specified as floating-point number 0.25 under *RatioNumerator*. The *RatioNumerator* may be negative.

Outputs

```
VAR_OUTPUT
    StartSync      : BOOL;
    InSync         : BOOL;
    Busy           : BOOL;
    Active         : BOOL;
    CommandAborted : BOOL;
    Error          : BOOL;
    ErrorID        : UDINT;
END_VAR
```

StartSync: Becomes TRUE when the synchronization with the master axis was started.

InSync: Becomes TRUE, if the coupling was successfully completed and the slave axis is synchronized with the master axis.

Busy: The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. If *Busy* becomes FALSE again, the function block is ready for a new job. At the same time one of the outputs *InSync*, *CommandAborted* or *Error* is set.

Active: Active indicates that the command is executed (currently Active=Busy, see BufferMode)

CommandAborted: Becomes TRUE, if the command could not be fully executed. The axis may have become decoupled during the coupling process (simultaneous command execution).

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number

Inputs/outputs

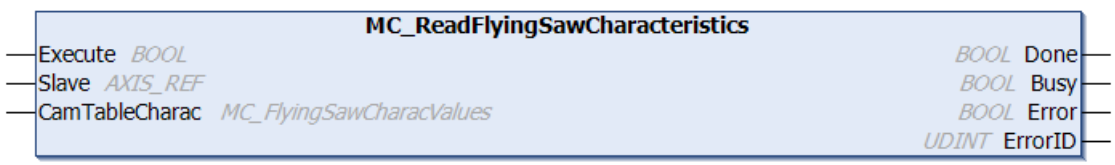
```
VAR_IN_OUT
    Master : AXIS_REF;
    Slave  : AXIS_REF;
END_VAR
```

Master: Master axis data structure.

Slave: Axis data structure of the Slave.

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

3.3 MC_ReadFlyingSawCharacteristics



The *MC_ReadFlyingSawCharacteristics* function allows the characteristic figures for the synchronization phase of the Universal Flying Saw to be read.

Inputs

```
VAR_INPUT
    Execute : BOOL;
END_VAR
```

Execute: A rising edge initiates reading the characteristic values from the TwinCAT NC. **Notice**: The data calculated is not available until the Universal Flying Saw starts.

Outputs

```
VAR_OUTPUT
    Done    : BOOL;
    Busy    : BOOL;
    Error   : BOOL;
    ErrorID : UDINT;
END_VAR
```

Done: Is set to TRUE when the data record has been successfully read.

Busy: The *Busy* output becomes TRUE when the command is started with *Execute* and remains TRUE as long as the command is processed. If *Busy* becomes FALSE again, the function block is ready for a new job. At the same time one of the outputs, *Done* or *Error*, is set.

Error: Becomes TRUE, as soon as an error occurs.

ErrorID: If the error output is set, this parameter supplies the error number

Inputs/outputs

```
VAR_IN_OUT
    Slave          : AXIS_REF;
    CamTableCharac : MC_FlyingSawCharacValues;
END_VAR
```

Slave: Axis structure of the slave.

CamTableCharac: Structure containing the characteristic values [▶ 12].

The axis data structure of type AXIS_REF addresses an axis uniquely within the system. Among other parameters it contains the current axis status, including position, velocity or error status.

4 Data types

4.1 The ST_SyncMode data structure

```
TYPE ST_SyncMode :
STRUCT
    (* mode *)
    GearInSyncMode                          : E_GearInSyncMode;

    (* 32 bit check mask ... *)
    GearInSync_CheckMask_MinPos             : BOOL;
    GearInSync_CheckMask_MaxPos             : BOOL;
    GearInSync_CheckMask_MaxVelo            : BOOL;
    GearInSync_CheckMask_MaxAcc             : BOOL;
    GearInSync_CheckMask_MaxDec             : BOOL;
    GearInSync_CheckMask_MaxJerk            : BOOL;
    GearInSync_CheckMask_OvershootPos       : BOOL;
    GearInSync_CheckMask_UndershootPos      : BOOL;
    GearInSync_CheckMask_OvershootVelo      : BOOL;
    GearInSync_CheckMask_UndershootVelo     : BOOL;
    GearInSync_CheckMask_OvershootVeloZero  : BOOL;
    GearInSync_CheckMask_UndershootVeloZero : BOOL;

    (* operation masks ... *)
    GearInSync_OpMask_RollbackLock          : BOOL;
    GearInSync_OpMask_InstantStopOnRollback : BOOL;
    GearInSync_OpMask_PreferConstVelo       : BOOL;
    GearInSync_OpMask_IgnoreMasterAcc       : BOOL;
END_STRUCT
END_TYPE

TYPE E_GearInSyncMode :
(
    (* synchronization based on the master position, slave dynamics depend on master dynamics *)
    GEARINSYNCMODE_POSITIONBASED,

    (* synchronization based on a standalone slave PTP profile, master independet slave dynamics *)
    GEARINSYNCMODE_TIMEBASED
);
END_TYPE
```

NOTE! The time-based motion profile (GEARINSYNCMODE_TIMEBASED) is currently only implemented for the function block MC_GearInVelo.

Operation of the individual bits

4.2 Data type MC_FlyingSawCharacValues

```
TYPE MC_FlyingSawCharacValues :
STRUCT
    (* Master Velocity*)
    fMasterVeloNom     : LREAL; (* 1. master nominal velocity (normed:=> 1.0) *)

    (* characteristic slave data *)
    (*===========================*)

    (* Start of cam table *)
    fMasterPosStart    : LREAL; (* 2. master start position*)
    fSlavePosStart     : LREAL; (* 3. slave start position *)
    fSlaveVeloStart    : LREAL; (* 4. slave start velocity *)
    fSlaveAccStart     : LREAL; (* 5. slave start acceleration *)
    fSlaveJerkStart    : LREAL; (* 6. slave start jerk *)


    (* End of cam table*)
    fMasterPosEnd      : LREAL; (* 7. master end position *)
    fSlavePosEnd       : LREAL; (* 8. slave end position *)
    fSlaveVeloEnd      : LREAL; (* 9. slave end velocity *)
    fSlaveAccEnd       : LREAL; (* 10. slave end acceleration *)
    fSlaveJerkEnd      : LREAL; (* 11. slave end jerk *)

    (* minimum slave position *)
    fMPosAtSPosMin     : LREAL; (* 12. master position AT slave minimum position *)
```

```
    fSlavePosMin       : LREAL; (* 13. slave minimum position *)

    (* minimum Slave velocity *)
    fMPosAtSVeloMin    : LREAL; (* 14. master position AT slave minimum velocity *)
    fSlaveVeloMin      : LREAL; (* 15. slave minimum velocity *)

    (* minimum slave acceleration *)
    fMPosAtSAccMin     : LREAL; (* 16. master position AT slave minimum acceleration *)
    fSlaveAccMin       : LREAL; (* 17. slave minimum acceleration *)
    fSVeloAtSAccMin    : LREAL; (* 18. slave velocity AT slave minimum acceleration *)

    (* minimum slave jerk and dynamic momentum *)
    fSlaveJerkMin      : LREAL; (* 19. slave minimum jerk *)
    fSlaveDynMomMin    : LREAL; (* 20. slave minimum dynamic momentum (NOT SUPPORTED YET !) *)

    (* maximum slave position *)
    fMPosAtSPosMax     : LREAL; (* 21. master position AT slave maximum position *)
    fSlavePosMax       : LREAL; (* 22. slave maximum position *)

    (* maximum Slave velocity *)
    fMPosAtSVeloMax    : LREAL; (* 23. master position AT slave maximum velocity *)
    fSlaveVeloMax      : LREAL; (* 24. slave maximum velocity *)

    (* maximum slave acceleration *)
    fMPosAtSAccMax     : LREAL; (* 25. master position AT slave maximum acceleration *)
    fSlaveAccMax       : LREAL; (* 26. slave maximum acceleration *)
    fSVeloAtSAccMax    : LREAL; (* 27. slave velocity AT slave maximum acceleration *)

    (* maximum Slave slave jerk and dynamic momentum *)
    fSlaveJerkMax      : LREAL; (* 28. slave maximum jerk *)
    fSlaveDynMomMax    : LREAL; (* 29. slave maximum dynamic momentum (NOT SUPPORTED YET !) *)

    (* mean and effective values *)
    fSlaveVeloMean     : LREAL; (* 30. slave mean absolute velocity (NOT SUPPORTED YET !) *)
    fSlaveAccEff       : LREAL; (* 31. slave effective acceleration (NOT SUPPORTED YET !) *)

    (* reserved space for future extension *)
    reserved           : ARRAY[32..47] OF LREAL;

    (* organization structure of the cam table *)
    CamTableID         : UDINT;
    NumberOfRows       : UDINT; (* number of cam table entries, e.g. number of points *)
    NumberOfColumns    : UDINT; (* number of table columns, typically 1 or 2 *)
    TableType          : UINT; (* MC_TableType *)
    Periodic           : BOOL;

    reserved2          : ARRAY[1..121] OF BYTE;
END_STRUCT
END_TYPE
```

Type definition for the characteristic parameters of a flying saw synchronization.

5 Example program

5.1 Flying saw sample program

The example program shows a typical cyclic flying saw sequence. The program manages three axes: master, slave and tool. The slave and tool axes are initially in their home position at 0. The master axis is started first. It can be regarded as a transport system that transports continuous material to a cutting unit. The material is to be cut at constant intervals. To this end the slave axis, i.e. the flying saw, is synchronised with the master axis and travels synchronous with the transport system. During this phase the tool axis is activated, which deals with the actual processing. The slave axis is then uncoupled and returns to its home position. During the return travel the next synchronisation process is started, with a point at a constant distance to the last machining position as target.

The example program requires the flying saw library and operates fully in simulation mode. Progress can be monitored in TwinCAT Scope View with the configuration provided.

Click here to save the example program:

http://infosys.beckhoff.com/content/1033/TF5055_TC3_NC_Flying_Saw/Resources/zip/2573794443.zip