

Manual | EN

TS5055

TwinCAT 2 | NC Flying Saw



Supplement | Motion

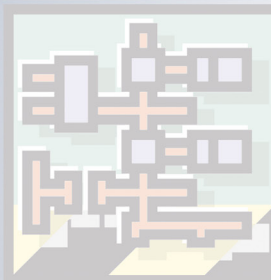


Table of contents

1 Foreword	5
1.1 Notes on the documentation	5
1.2 Safety instructions	6
1.3 Notes on information security.....	7
2 General	8
3 Synchronisation to velocity	11
4 Synchronisation to position	15
5 Parameterisable boundary conditions, specifying the mode of operation	18
6 Characteristic values	21
7 Calculating the synchronisation phase	23
8 Reversal of the master axis movement / reverse motion stop	25
9 Diagonal saw	29
10 Interfaces	30
11 Operation from the System Manager	32
12 PLC API	33
12.1 TcMC2_FlyingSaw	33
12.1.1 MC_GearInVelo	33
12.1.2 MC_GearInPos	36
12.1.3 MC_ReadFlyingSawCharacteristics.....	39
12.2 TcMC2.....	40
12.2.1 MC_GearOut.....	40
12.3 Data types	41
12.3.1 ST_SyncMode.....	41
12.3.2 MC_FlyingSawCharacValues	42
12.4 Example program	44
12.4.1 Flying saw sample program	44
13 Error situations and error codes:	45

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 applicable national standards.

It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.

It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

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.

We reserve the right to revise and change the documentation at any time and without prior announcement. 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®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® 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, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT®

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited.

Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

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.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.



Tip or pointer

This symbol indicates information that contributes to better understanding.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our <https://www.beckhoff.com/secguide>.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

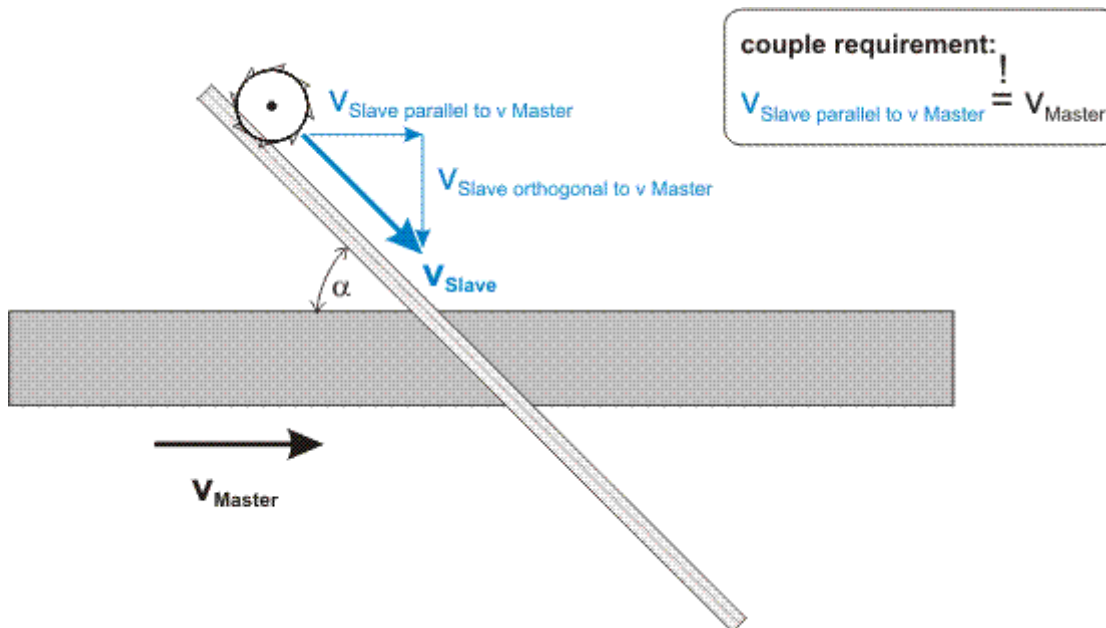
To stay informed about information security for Beckhoff products, subscribe to the RSS feed at <https://www.beckhoff.com/secinfo>.

2 General

The Flying Saw is a slave axis that can be synchronized to a moving master axis. The slave axis moves in synchronism with the master axis to perform machining processes. This kind of movement, synchronized to the master axis, means that a workpiece can be machined even while it is being transported.

An important difference between the "Flying Saw" and the "Universal Flying Saw" is associated with the initial conditions required of the slave axis for the synchronization. The "Universal Flying Saw", unlike the "Flying Saw", is able to start synchronization of the slave even when the slave has already started, and is therefore no longer stationary. The "Universal Flying Saw" also calculates improved set value profiles, and these can be influenced by the user through a wide range of boundary conditions.

The ratio of the master velocity to the slave velocity in the synchronous phase is parameterized via a variable coupling factor. This coupling factor in the case of a diagonal saw, for instance, is chosen to be unequal to 1, so that the velocity component of the slave axis in the direction of the master axis movement ($v_{\text{slave parallel to } v_{\text{master}}}$) in the synchronized phase is equal to the master velocity (v_{master}). (See diagram.)



The Universal Flying Saw basically provides two different synchronization methods. In the case of *synchronization to velocity* the slave is synchronized to the master as quickly as possible, bearing in mind the coupling factor. The coupling position for the master and slave axes therefore results from having set the fastest possible synchronization as the target. In contrast to this, the coupling position of the master and slave axes is parameterized by the user under *synchronization to position*. The master and slave movements will in this case therefore be moving in synchronization as from the specified position at the latest.

Both of these synchronization methods permit a variety of boundary conditions to be specified for the synchronization phase. These boundary conditions make it possible to adapt the synchronization process to the needs of the machine.



This manual describes the Universal Flying Saw *TcMc2_FlyingSaw.lib*, which is available from TwinCAT Version 2.9, Build 248. If you are using the previous version *TcNcFlyingSaw.lib* and need further information, see [here](#).

Interfaces

The Universal Flying Saw is operated and monitored from the PLC using appropriate function blocks. For commissioning purposes, however, the Universal Flying Saw can also be started directly from the [TwinCAT System Manager](#) [► 32]. In this case, the cyclic NC/PLC axis interface and ADS communication are used as the underlying interface.

Synchronisation to velocity

In [Synchronisation to velocity \[► 11\]](#) the slave axis is synchronised to the master axis using the specified dynamic parameters as rapidly as possible. In the synchronous phase, the slave velocity is proportional to the master velocity, so that:

$$v_{Slave} = F_{CouplingFactor} \cdot v_{Master}$$

The synchronisation procedure

Synchronisation of the slave axis to the master axis proceeds according to the following scheme:

1. Starting the Universal Flying Saw. This corresponds to the logical coupling to the master axis. This moment is referred to as the coupling time.
2. The synchronisation phase: The slave is accelerated from its initial condition up to the velocity of the master whilst observing the boundary conditions for slave movement specified by the user. The time at which the synchronisation phase changes to the synchronous phase is referred to as the synchronisation time.
3. Synchronous phase: The slave moves synchronously with the master.
4. Uncoupling the Universal Flying Saw. This is an online change. The coupled slave once again becomes an independent master that continues to move without limit with the velocity resulting from the online change.
5. This could mean that the former slave restarts or stops. The full functionality of a TwinCAT NC master axis is once more available.

Synchronisation to position

In [Synchronisation to position \[► 15\]](#) the slave axis is synchronised to the master at the specified synchronisation position using the specified dynamic parameters. This means that the slave axis reaches the synchronous velocity at exactly the synchronisation position of master and slave, after which it moves in synchronism with the master. The slave velocity in the synchronous phase is governed by:

$$v_{Slave} = F_{CouplingFactor} \cdot v_{Master}$$

The synchronisation procedure

Synchronisation of the slave axis to the master axis proceeds according to the following scheme:

1. The start of the Universal Flying Saw. This is the logical coupling to the master axis. This moment is referred to as the coupling time.
2. The synchronisation phase: The slave is accelerated from its initial condition to the master's velocity, reaching the slave synchronisation position and synchronisation velocity precisely at the specified master synchronisation position. The boundary conditions specified by the user for slave movement are maintained during this process. The time at which the synchronisation phase changes to the synchronous phase is referred to as the synchronisation time.
3. Synchronous phase: The slave moves synchronously with the master.
4. Uncoupling the Universal Flying Saw. This is an online change. The coupled slave once again becomes an independent master that continues to move without limit with the velocity resulting from the online change.
5. This could mean that the former slave restarts or stops. The full functionality of a TwinCAT NC master axis is once more available.

Parameterisable boundary conditions governing synchronisation

In principle, any initial conditions may apply to the calculation of the synchronisation profile for the master and slave axes.

The transition of the slave's movement from its initial state to the synchronous state is calculated in such a way that boundary conditions [► 18] that can be specified by the user and that govern the slave's movement are maintained. These boundary conditions can be used, for instance, to limit the maximum slave velocity, or to prevent an overshoot in its position.

The calculation and checking of the parameterisable boundary conditions proceeds on the basis of the characteristic values determined for the synchronisation phase. In the determination of the *characteristic values*, the idealised assumption is made that the master axis will continue to move at a constant velocity, i.e. with no acceleration, after the coupling time. Exact calculation and checking of the parameterisable boundary conditions is only possible if this assumption is made. Any other reasonable assumption about the future movement of the master is not possible, since the master's future movement is not known at the time of coupling.

An acceleration of the master that might occur in the future will also affect the slave dynamics as a result of the coupling. Such acceleration by the master will have the effect that the calculated and checked values may be overshoot or undershot in some cases, depending on the master's acceleration. Characteristic values that may be affected by master acceleration can be seen in the tabular description of the characteristic values [► 21].

Characteristic values describing slave movement

The characteristic values [► 21] governing the movement that the slave will undergo during the synchronisation phase are available to the user after the Universal Flying Saw has been started. This value structure contains magnitudes such as the maximum slave acceleration, the minimum and maximum slave position, and so forth. These values are calculated under the assumption that the master is free from acceleration, and are therefore in some cases only exactly correct for such a case.

NOTE

The master acceleration at the time that the "Universal Flying Saw" starts has a significant effect on the profile calculation and its optimization. This means that if an encoder axis is the master, the velocity and the acceleration must be carefully filtered, or even the calculation of the actual acceleration must be deselected (see. "Encoder mode").

3 Synchronisation to velocity

Under synchronization to velocity, a slave axis, starting in any state, is synchronized as quickly as possible to a moving master axis. The synchronous velocity of the slave here is given by the master velocity multiplied by the coupling factor.

The synchronization phase of the slave axis is calculated in such a way that the [boundary conditions \[18\]](#) specified by the user are maintained. Calculation and checking of these boundary conditions is carried out under the assumption that the master axis will continue to move without acceleration after the coupling time. If the master axis is not free moving, some overshoot or undershoot of the parameterized boundary conditions may occur.

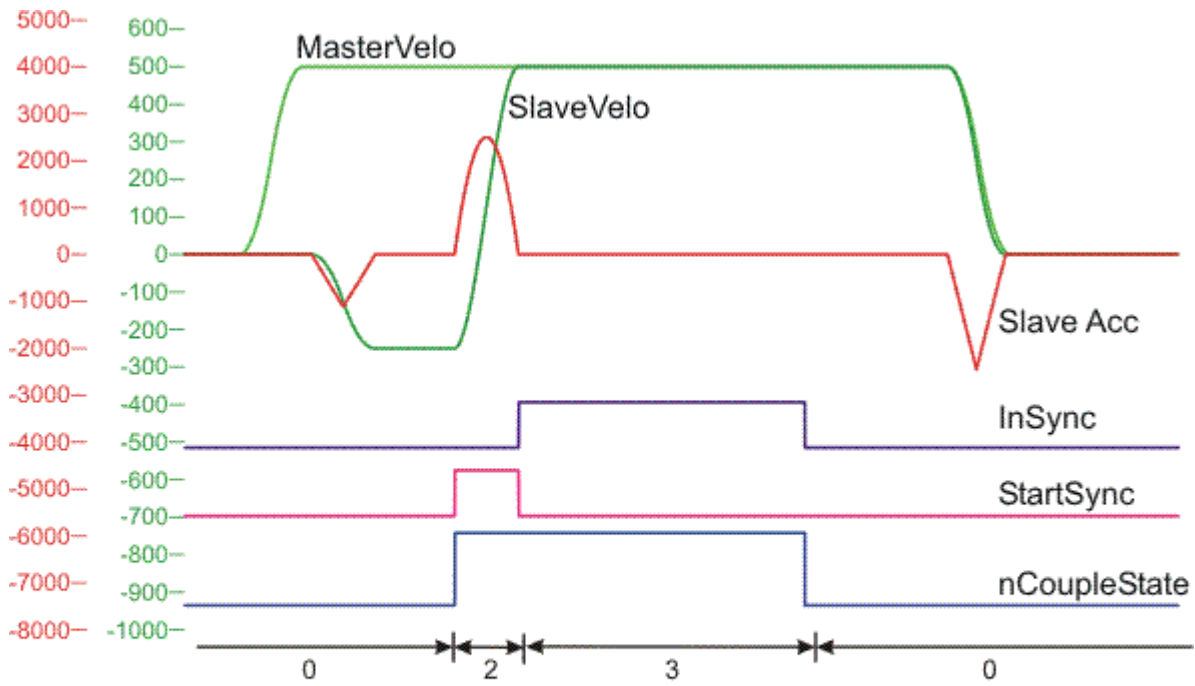
Example 1:

In the illustrated phase 0 (left) the two axes are moving entirely independently. The future master axis is accelerating to 500 mm/s while the future slave axis is accelerating to -250 mm/s. As the phase changes from 0 to 2 the process of synchronization to velocity in the Universal Flying Saw starts (`StartSync = 1`) and the set value profile for the synchronization is calculated. This calculated set value profile for the slave is then specified for phase 2 of the slave axis. At the end of phase 2 the slave axis precisely achieves the synchronous velocity, which is the velocity of the master axis multiplied by the coupling factor. From this time on the axes are in the synchronous phase (phase 3). In this synchronous phase the two axes move synchronously in accordance with the coupling factor. The synchronous phase is ended by the uncoupling command. In the illustration, this corresponds to the transition from phase 3 to phase 0. From this time on they are again two independent master axes. The slave axis changes to a master axis online at the uncoupling time. This online change will remove any acceleration or deceleration to which the slave may be subject, thus fixing the velocity of the former slave, with which it will then continue to move without limit.

The phase currently applying to the slave axis can be seen in the `nAxisState` variable in the cyclic axis interface. (The names of the individual phases in the illustrations do not agree with the values of the [nAxisState \[30\]](#) variable.)

```
(* Start parameters *)
fMasterVelo    := 500;
fSlaveVelo     := -250;

(* Coupling parameters *)
fGearRatio     := 1.0;
fSlaveAcc      := 2500;
fSlaveDec      := 2500;
fSlaveJerk     := 5000;
```



phase 0: independent master axes

phase 2: synchronisation

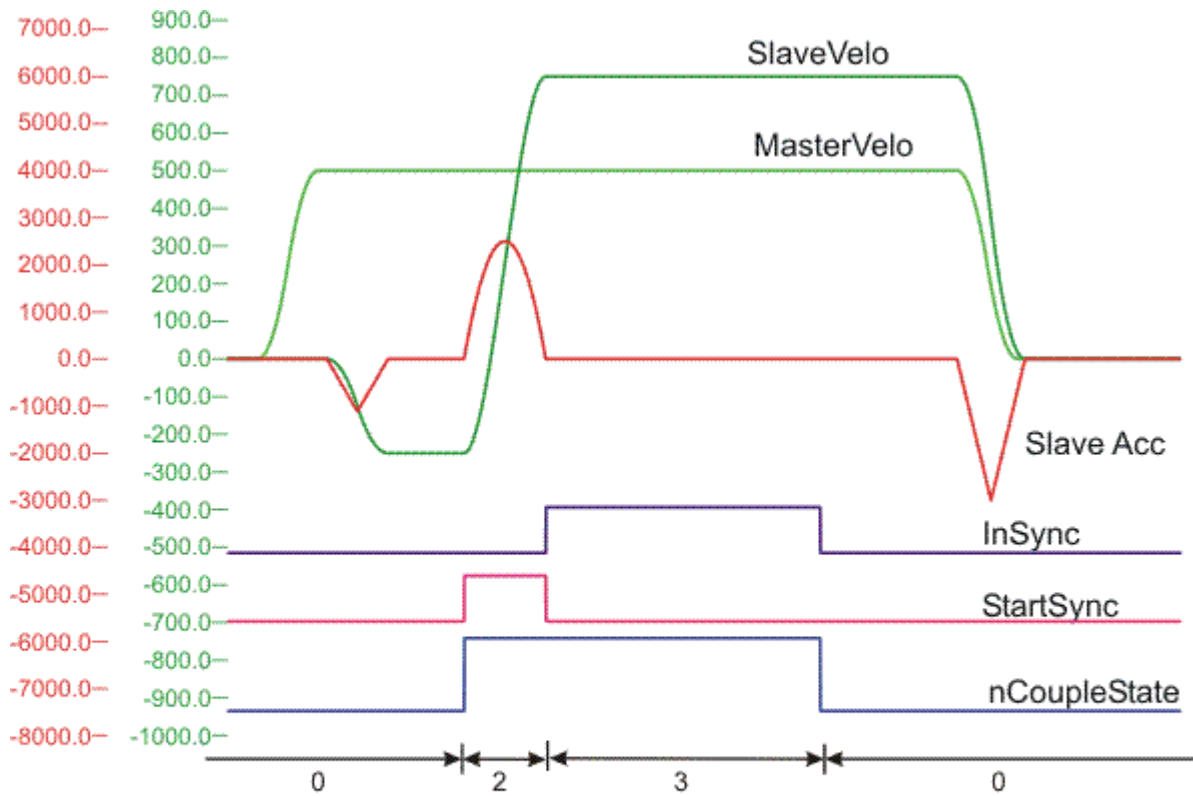
phase 3: synchronous phase

Example 2:

As example 1, but with coupling factor 1.5.

```
(* Start parameters *)
fMasterVelo := 500;
fSlaveVelo  := -250;

(* Coupling parameters *)
fGearRatio  := 1.5;
fSlaveAcc   := 2500;
fSlaveDec   := 2500;
fSlaveJerk  := 5000;
```



phase 0: independent master axes

phase 2: synchronisation

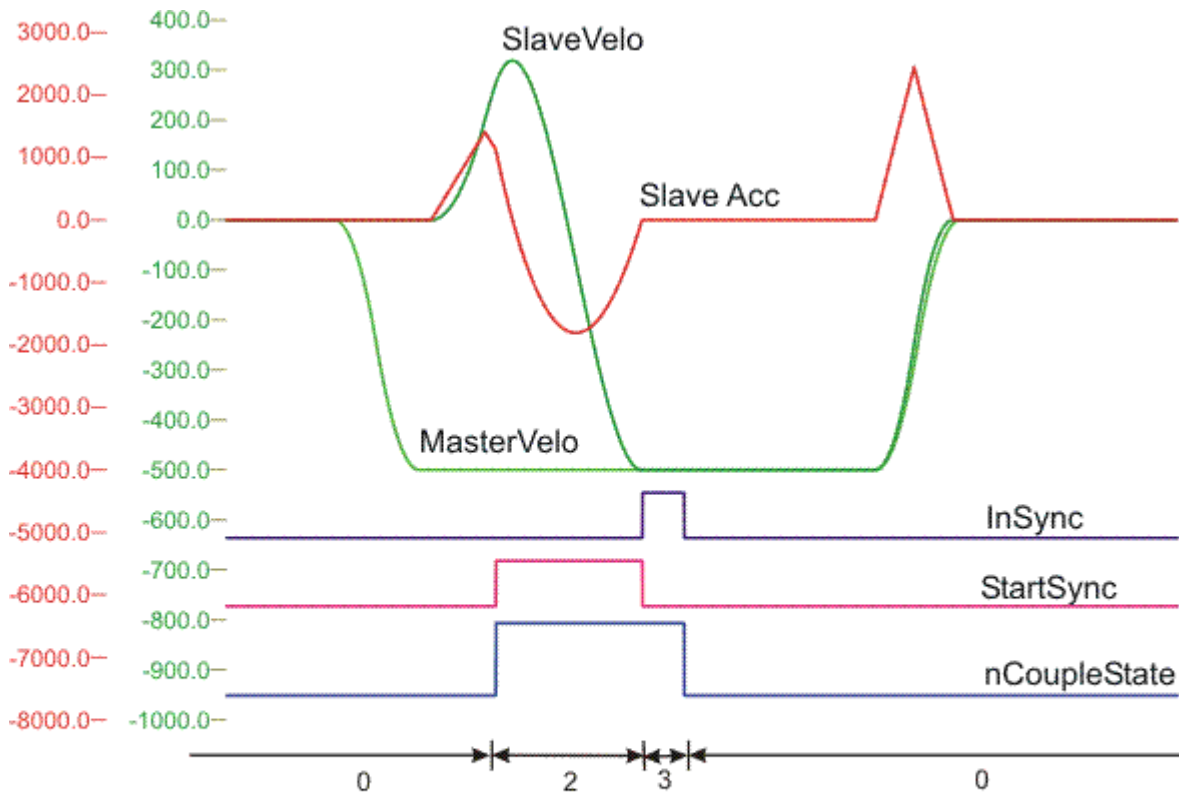
phase 3: synchronous pahse

Example 3:

The start of the coupling in the acceleration phase of the future slave.

```
(* Start parameters *)
fMasterVelo := -500;
fSlaveVelo := 400;

(* Coupling parameters *)
fGearRatio := 1;
fSlaveAcc := 2500;
fSlaveDec := 2500;
fSlaveJerk := 5000;
```



phase 0: independent master axes

phase 2: synchronisation

phase 3: synchronous phase

PLC function blocks

The function block MC_GearInVelo is used for the coupling. To end the synchronous phase by uncoupling (online change) the function block MC_GearOut is used.

4 Synchronisation to position

In the case of synchronisation to position, the slave axis, starting from its initial state, is synchronised to the master axis in such a way that the required synchronous velocity ($v_{\text{master}} \cdot \text{coupling factor}$) and synchronisation position are achieved precisely at the master's synchronisation position.

The synchronisation phase of the slave axis is calculated in such a way that the boundary conditions [► 18] specified by the user are maintained. Calculation and checking of these boundary conditions is carried out under the assumption that the master axis will continue to move without acceleration after the coupling time. If the master axis is not free moving, some overshoot or undershoot of the parameterised boundary conditions may occur.

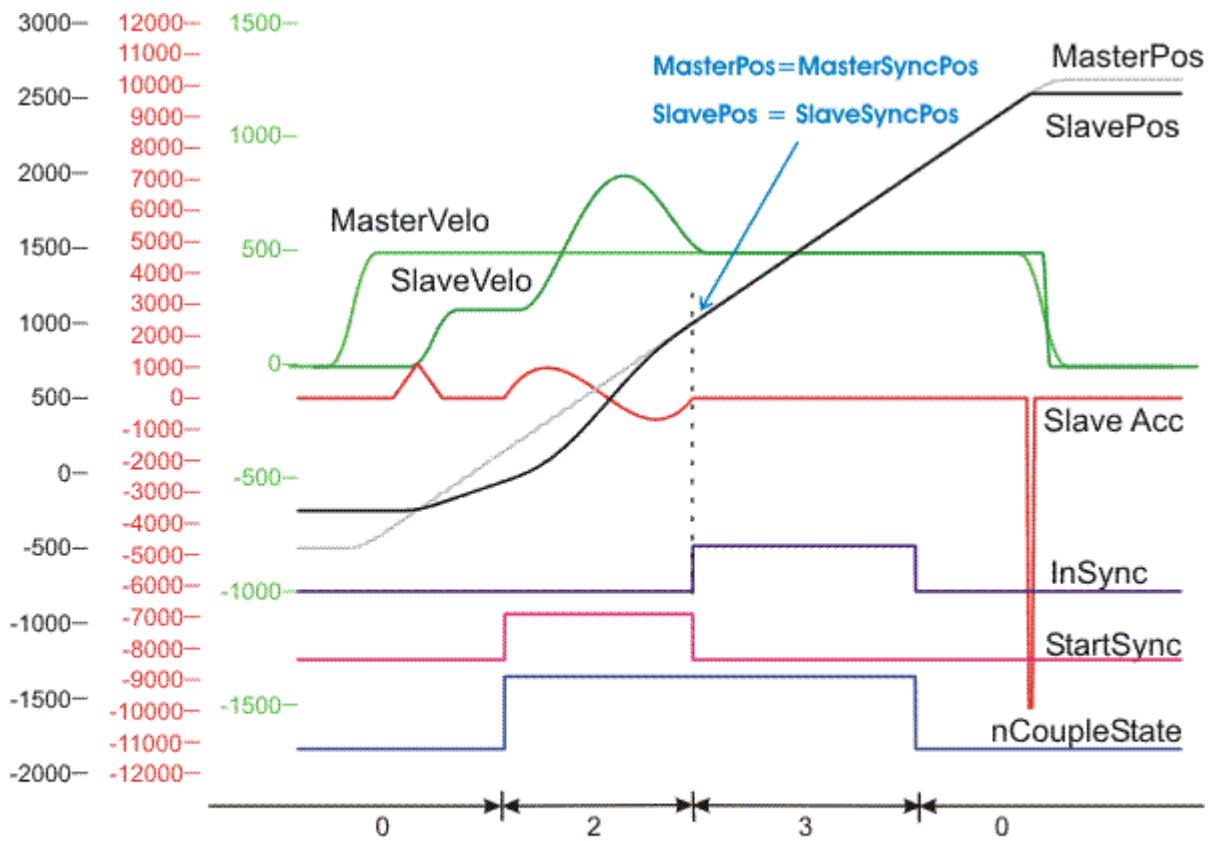
Example 1:

In the example illustrated, the future master and slave axes are independently started, moving in positive directions. At the time of the illustrated phase change from 0 to 2, the Universal Flying Saw is started with synchronisation to position (StartSync = 1). Synchronisation to position means that the slave axis reaches the synchronous velocity precisely as the master is at the master synchronisation position (master-sync-position) and when the slave is at the slave synchronisation position (slave-sync-position). The synchronous velocity corresponds to the master velocity multiplied by the chosen coupling factor. In the example illustrated here, the master-sync-position is identical to the slave-sync-position, which means that the master and slave positions are the same when the *InSync* signal provides a rising edge. In this example, 1 has been selected as the coupling factor, so that the master and slave velocities in the synchronous phase (3) are identical. At the time when the phase changes from 3 to 0, the slave axis is uncoupled from the master axis (online change), and then continues to move once again as an independent master axis. The slave axis changes to a master axis online at the uncoupling time. This online change will remove any acceleration or deceleration to which the slave may be subject, thus fixing the velocity of the former slave, with which it will then continue to move without limit.

The phase currently applying to the slave axis can be seen in the `nAxisState` variable in the cyclic axis interface. (The names of the individual phases in the illustrations do not agree with the values of the `nAxisState` [► 30] variable.)

```
(* Start parameters *)
fMasterVelo      := 500;
fSlaveVelo       := 250;
fMasterStartPos  := -500;
fSlaveStartPos   := -250;

(* Coupling parameters *)
fGearRatio       := 1;
fMasterSynchronPos := 1000;
fSlaveSynchronPos := 1000;
fSlaveAcc        := 2500;
fSlaveDec        := 2500;
fSlaveJerk       := 5000;
```



phase 0: independent master axes

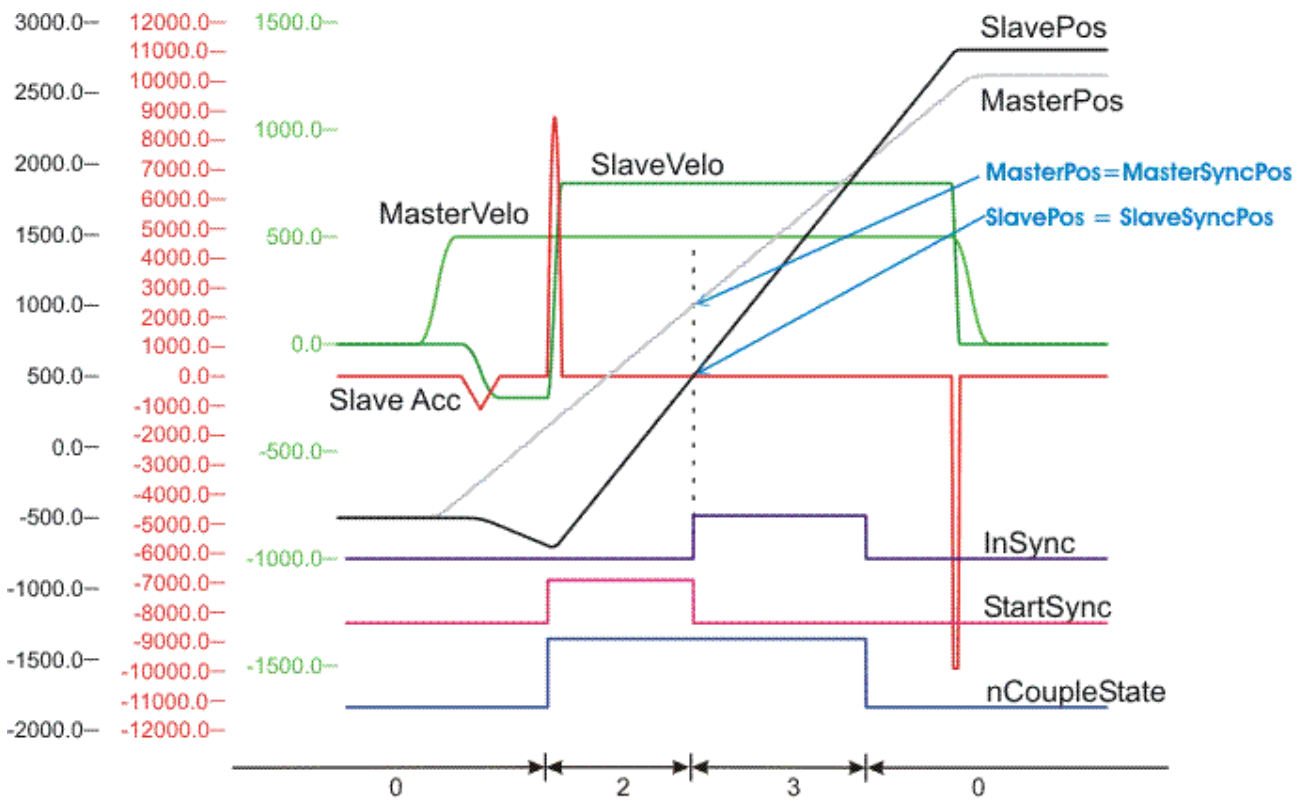
phase 2: synchronisation

phase 3: synchronous phase

Example 2:

```
(* Start parameters *)
fMasterVelo      := 500;
fSlaveVelo      := -250;
fMasterStartPos := -500;
fSlaveStartPos  := -500;

(* Coupling parameters *)
fGearRatio      := 1.5;
fMasterSynchronPos := 1000;
fSlaveSynchronPos  := 500;
fSlaveAcc        := 10000;
fSlaveDec        := 10000;
fSlaveJerk       := 50000;
```

PLC function blocks

The function block MC_GearInPos is used for coupling. To end the synchronous phase, i.e. for uncoupling (online change of the slave into an independent master), the function block MC_GearOut is used.

5 Parameterisable boundary conditions, specifying the mode of operation

It is possible to specify a wide variety of boundary conditions for the slave movement in the synchronization phase of the Universal Flying Saw. These boundary conditions make it possible to specify limit values for the slave magnitudes listed in the table below. The *SyncMode* bit mask can be used to check whether the individual limit values are being observed. The boundary conditions specified for the synchronization phase also affect the set value profile for the synchronization. Whether, and in what way, the conditions affect the profile can be seen in the [diagram \[► 23\]](#) in the chapter below.

Synchronization mode			Description	
Define:				
GealnSyncMode				
GEARINSYNCMODE_POSITIONBASED			In this mode of the universal flying saw, a profile dependent on the master position is generated to synchronize the slave axis to the master axis.	
GEARINSYNCMODE_TIMEBASED			In this mode of the universal flying saw, a time-dependent motion profile is generated for synchronizing the slave axis to the master axis, which ensures compliance with all dynamic limit values of the slave axis. This mode is currently only available with coupling on velocity.	
Bit masks for the "SyncMode"			Description	Boundary condition
Define:	Value Decimal	Value Hexadecimal		
GEARINSYNC_CH ECKMASK_				
MINPOS	1	0x0000 0001	Checks whether the slave axis has passed below its software minimum end position (machine data).	$pos_{Slave} \geq pos_{SlaveMin}$
MAXPOS	2	0x0000 0002	Checks whether the software maximum end position (machine data) of the slave axis has been exceeded.	$pos_{Slave} \leq pos_{SlaveMax}$
MAXVELO	4	0x0000 0004	Checks whether the maximum permitted slave velocity (machine data) has been exceeded.	$ v_{Slave} \leq v_{SlaveMax}$
MAXACC	8	0x0000 0008	Checks whether the maximum slave acceleration (machine data) has been exceeded	$acc_{Slave} \leq acc_{SlaveMax}$
MAXDEC	16	0x0000 0010	Checks whether the maximum slave deceleration (machine data) has been exceeded	$dec_{Slave} \leq dec_{SlaveMax}$

MAXJERK	32	0x0000 0020	Checks whether the maximum slave jerk (machine data) has been exceeded.	$j_{Slave} \leq j_{SlaveMax}$
OVERSHOOTPOS	256	0x0000 0100	Checks for overshooting of slave position.	
UNDERSHOOTPOS	512	0x0000 0200	Checks for undershooting of slave position.	
OVERSHOOTVELO	1024	0x0000 0400	Checks for overshooting of slave acceleration.	
UNDERSHOOTVELO	2048	0x0000 0800	Checks for undershooting of slave acceleration.	
OVERSHOOTVELOZERO	4096	0x0000 1000	Checks whether the slave velocity has exceeded 0.0.	
UNDERSHOOTVELOZERO	8192	0x0000 2000	Checks whether the slave velocity is below 0.0	
Bit masks for operation modes		Description		
GEARINSYNC_OPMASK_				
ROLLBACKLOCK	65536	0x0001 0000	Bit = 0: (default) When the slave has achieved the synchronous phase, synchronous coupling of all the following master movements is maintained until the coupling is removed. This also applies if the master changes direction and moves backwards over the coupling position. (Further explanations) [▶ 25]	
			Bit = 1: Setting this bit activates the backstop, which causes the slave to stop when the master moves backwards beyond the coupling position after a motion reversal. (Further explanations) [▶ 25]	
INSTANTSTOPONROLLBACK	131072	0x0002 0000	Bit = 0: (default) On reaching the coupling position, the slave velocity is reduced smoothly following a 5th order polynomial. The polynomial is optimized to halt the slave as quickly as possible. (Further explanations) [▶ 25]	
			Bit = 1: In terms of the set value, the slave is halted within one NC tick of reaching the coupling position. The slave velocity is set to 0.0 and the position is maintained. This abrupt stop can trigger the following error monitoring system! (Further explanations) [▶ 25]	

<p>PREFERCONSTV ELO</p>	<p>1048576</p>	<p>0x0010 0000</p>	<p>Bit = 0: (default) Default setting</p> <hr/> <p>Bit = 1: The system will try to use a phase with constant velocity, instead of just one 5th order polynomial. This can result in a combination of a 5th-order polynomial, a synchronous phase and another 5th-order polynomial (P5-P1-P5), (see further explanation) [▶ 25]. The maximum given acceleration and deceleration is used. To control and limit the jerk it is recommended to set MAXJERK in the bit mask.</p>
<p>IGNOREMASTER ACC</p>	<p>2097152</p>	<p>0x0020 0000</p>	<p>Bit = 0: (default) Default setting</p> <hr/> <p>Bit = 1: When calculating the coupling, the acceleration of the master is ignored, i.e. set to zero. This causes the use of internal optimizations. At moderate acceleration, this specification leads to tolerable following errors. After the following error has been reduced, the relative position accuracy is independent of this setting.</p>

NOTE

The checks listed above apply only to the synchronization phase (GEARINSYNCSTATE_SYNCHRONIZING), not to the phase of synchronized movement. These calculations and checks are also only possible when the assumption is made that the master continues to move with constant velocity after the coupling time, i.e. that it is not subject to acceleration. Making other assumptions for the master makes no sense, since at the time of coupling it is generally not known how the master will move in the future.

6 Characteristic values

Once the Universal Flying Saw has been started, the MC_ReadFlyingSawCharacteristics PLC function block can be used to read the characteristic values associated with the slave set value profile that has been calculated for the **synchronisation phase**. These characteristic values therefore refer neither to the synchronous phase nor to the preliminary phase, but only apply to the synchronisation phase!

The individual characteristic values are described in the table below.

NOTE

When calculating the dynamic characteristic values it is assumed that after the Universal Flying Saw has been started (after the coupling moment) the master will continue to move without acceleration. It follows that the calculated characteristic values are only precisely correct for a master that is not accelerating, and that in practice there may be overshoot or undershoot if the master accelerates or decelerates.

Structure of the characteristic values

The characteristic values are stored in the comprehensive *MC_FlyingSawCharacValues* structure, which contains the following magnitudes:



The start time described in the table always refers to the start of the synchronisation phase, while the end time refers to the end of the synchronisation phase (not the end of the synchronous phase)! The focus here is always on the synchronisation phase.

The third column of the table indicates whether any acceleration that the master may undergo will affect the characteristic value or not.

Denomination	Description	Independent of master acceleration
fMasterVeloNom	Master velocity at the start time of the Universal Flying Saw	no
fMasterPosStart	Master position at the start time of the Universal Flying Saw	yes
fSlavePosStart	Slave position at the start time of the Universal Flying Saw	yes
fSlaveVeloStart	Slave velocity at the start time of the Universal Flying Saw	no
fSlaveAccStart	Slave acceleration at the start time of the Universal Flying Saw	no
fSlaveJerkStart	Slave jerk at the start time of the Universal Flying Saw	no
fMasterPosEnd	Master position at the end of the synchronisation phase	yes
fSlavePosEnd	Slave position at the end of the synchronisation phase	yes
fSlaveVeloEnd	Slave velocity at the end of the synchronisation phase	no
fSlaveAccEnd	Slave acceleration at the end of the synchronisation phase	no
fSlaveJerkEnd	Slave jerk at the end of the synchronisation phase	no
fMPosAtSPosMin	Master position at the time of the minimum slave position	no

Denomination	Description	Independent of master acceleration
fSlavePosMin	Minimum slave position	yes
fMPosAtSVeloMin	Master position at the time of the minimum slave velocity	no
fSlaveVeloMin	Minimum slave velocity	no
fMPosAtSAccMin	Master position at the time of the minimum slave acceleration	no
fSlaveAccMin	Minimum slave acceleration	no
fSVeloAtSAccMin	Slave velocity at the time of the minimum slave acceleration	no
fSlaveJerkMin	Minimum slave jerk	no
fSlaveDynMomMin	Minimum slave dynamic moment (NOT SUPPORTED YET!)	no
fMPosAtSPosMax	Master position at the time of the maximum slave position	no
fSlavePosMax	Maximum slave position	yes
fMPosAtSVeloMax	Master position at the time of the maximum slave velocity	no
fSlaveVeloMax	Maximum slave velocity	no
fMPosAtSAccMax	Master position at the time of the maximum slave acceleration	no
fSlaveAccMax	Maximum slave acceleration	no
fSVeloAtSAccMax	Slave velocity at the time of the maximum slave acceleration	no
fSlaveJerkMax	Maximum slave jerk	no
fSlaveDynMomMax	Minimum slave dynamic moment (NOT SUPPORTED YET!)	no
fSlaveVeloMean	Mean absolute slave velocity (NOT SUPPORTED YET !)	no
fSlaveAccEff	Effective slave acceleration (NOT SUPPORTED YET !)	no

7 Calculating the synchronisation phase

An attempt is made when calculating the synchronisation phase to find an optimum solution while observing the boundary conditions specified by the user. If it is not possible to observe the specified boundary conditions, the coupling is declined and an appropriate error message is issued.

Optimizations

As can be seen in the flow chart below, the individual bit masks partly influence the internal optimization steps of the profile calculation, since an optimum is searched for depending on predefined rules (see [Parameterizable boundary conditions](#) [► 18]). Essentially, a 5th order polynomial or a combination of a 5th order polynomial with a 1st order polynomial is used. A 5th order polynomial is generally not free from overshoot, but the accelerations are more moderate than when combining a 5th order polynomial with a 1st order polynomial. The combination of the 1st and 5th order polynomials is calculated in such a way that it is always free from overshoot. However, higher accelerations and decelerations occur with it. If, for example, the actual velocity matches the synchronous velocity, but a certain position difference must be compensated. Then the optimum velocity is calculated internally as a function of the maximum acceleration. Result is a 5th power polynomial a 1st order polynomial with the calculated velocity and a 5th power polynomial. At least one of the two 5th power polynomials exploits the maximum acceleration. To avoid extreme jerk values, the jerk check should be switched on.

i The optimizations shown can only be carried out if both the master and slave axes are free of acceleration at the time of coupling. For accelerated axes, a 5th order polynomial is used for synchronization, which is checked for compliance with the specified boundary conditions, but cannot be optimized.

NOTE

If the master axis is an encoder axis (an "external encoder system"), something which as a rule is never mathematically free from acceleration, particular care must be taken to filter the actual acceleration. Alternatively the determination of the actual acceleration can be deselected in the encoder, i.e. set to zero. The NC also has an internal algorithm for this combination (master encoder axis with the Universal Flying Saw as a slave). This algorithm sets master accelerations whose magnitudes are less than $(2.0 \cdot \text{scaling factor} / \text{cycle time}^2)$ to zero at the coupling time.

Optimisation step 1:

Aim: "Velocity profile free from undershoot or overshoot"

An attempt is first made to calculate a profile that synchronises the velocity without overshoot or undershoot (a combination of a first order polynomial and a 5th order polynomial or vice versa: in abbreviated form, polynomial1+polynomial5 or polynomial5+polynomial1). If the acceleration check is active at this stage, and if one or more of these limit values (acc, dec) is exceeded, then another profile, which in general is not free from overshoot (polynomial5) is calculated. If one of the active limit values (acc, dec) is still exceeded with his profile, then the synchronisation command is finally declined with an error code.

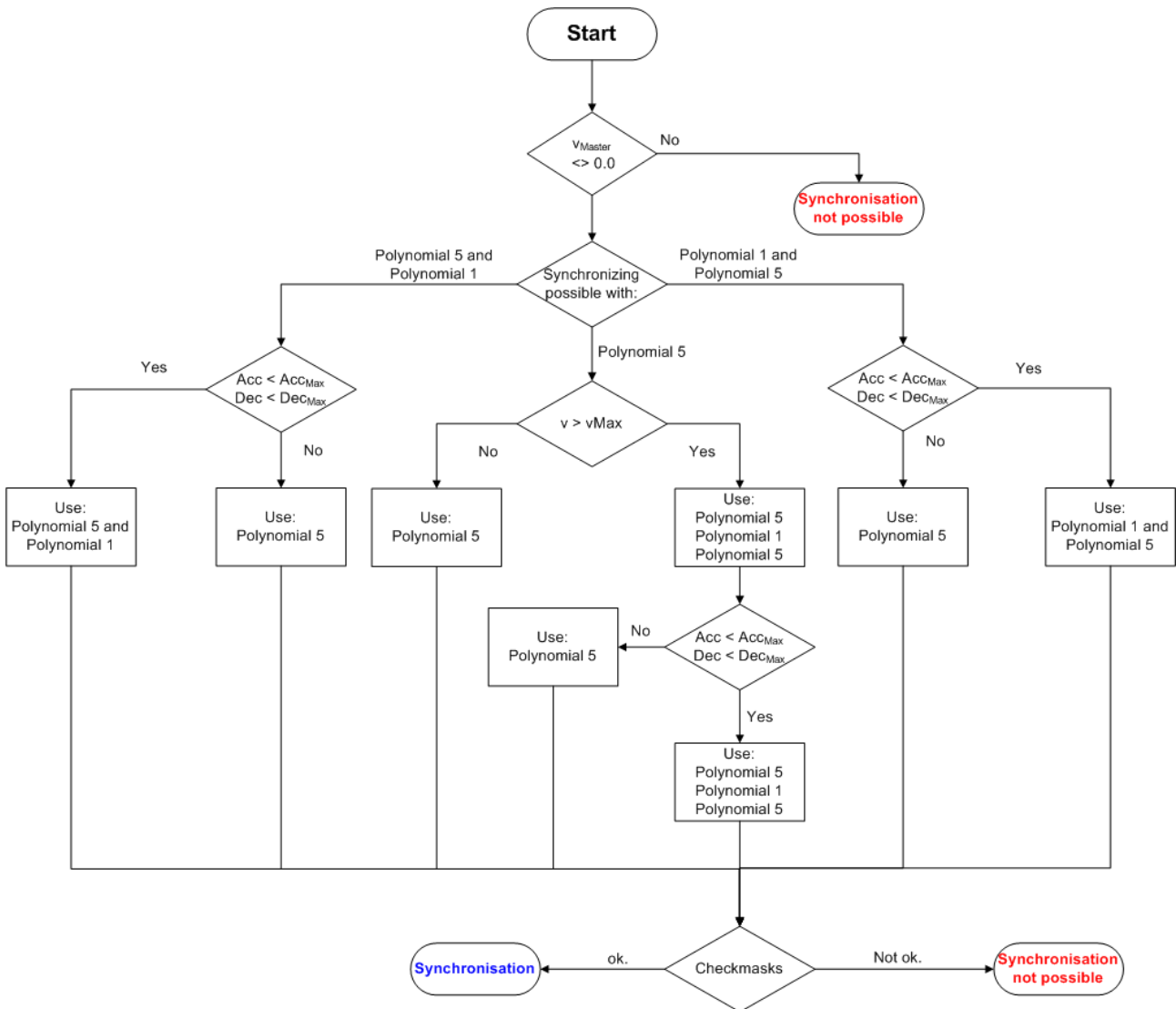
Optimisation step 2:

Aim: "Limitation to maximum permitted velocity"

If the first optimization step is not possible, the second optimization step checks whether the maximum permitted velocity for the slave axis is exceeded by a general standard profile (polynomial5). If this is the case, an attempt is made to generate a profile in which the maximum profile velocity is precisely the maximum velocity permitted to the slave axis (machine data) (polynomial5+polynomial1+polynomial5). It should be noted that this optimization attempt usually results in larger values of acceleration or deceleration. If the acceleration check is active at this stage, and if one or more of these limit values is exceeded, then this second optimization step is rejected, and finally a profile, which is not in general free from overshoot (polynomial5), is calculated. If one of the active limit values (acc, dec) is still exceeded with his profile, then the synchronization command is finally declined with an error code.

Optimisation flow diagram:

The optimizations carried out internally are illustrated in the following flow diagram. Essentially, the slave set value profile is calculated as a 5th order polynomial. This 5th order polynomial can be combined with a first order polynomial in order to maintain the parameterized boundary conditions. The way in which the individual boundary conditions influence the selection of the polynomials, and therefore the form of the set value profile, can be seen from the flow diagram illustrated below. The label "polynomial n and polynomial m" expresses the fact that polynomial n is first used in the synchronization phase, followed by polynomial m.



8 Reversal of the master axis movement / reverse motion stop

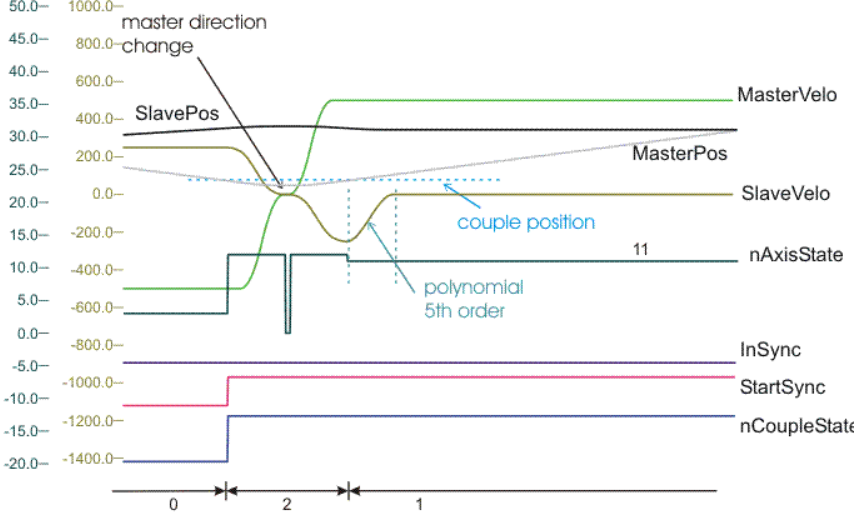
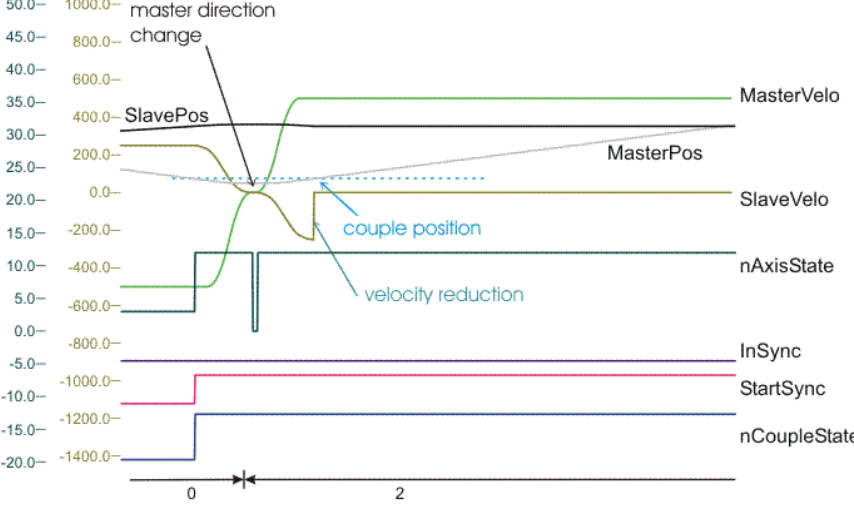
The behavior of the Universal Flying Saw in case of a motion reversal of the master can be defined via 2 bits of the SyncMode. The *GEARINSYNC_OPMASK_ROLLBACKLOCK* bit activates the backstop, which immobilizes the slave if the master moves backwards beyond the coupling position (the position where the Universal Flying Saw was started) after a motion reversal. The second bit, *GEARINSYNC_OPMASK_INSTANTSTOPONROLLBACK*, governs dynamic aspects of how the slave comes to a halt.

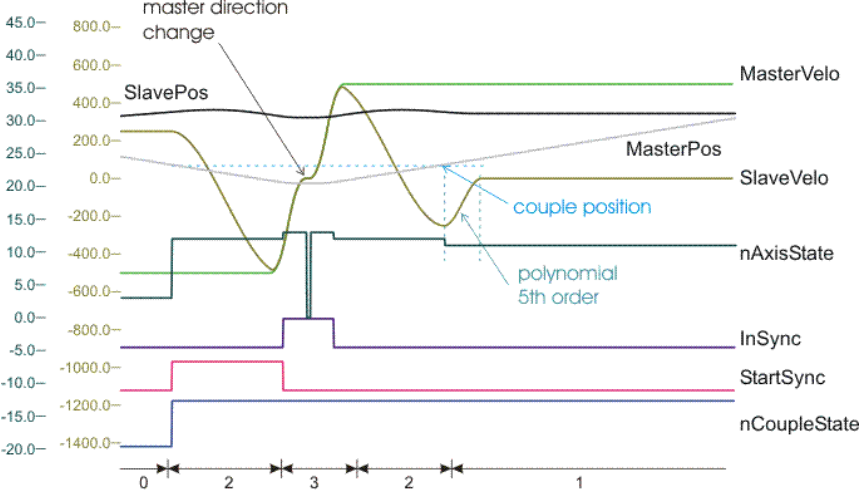
NOTE

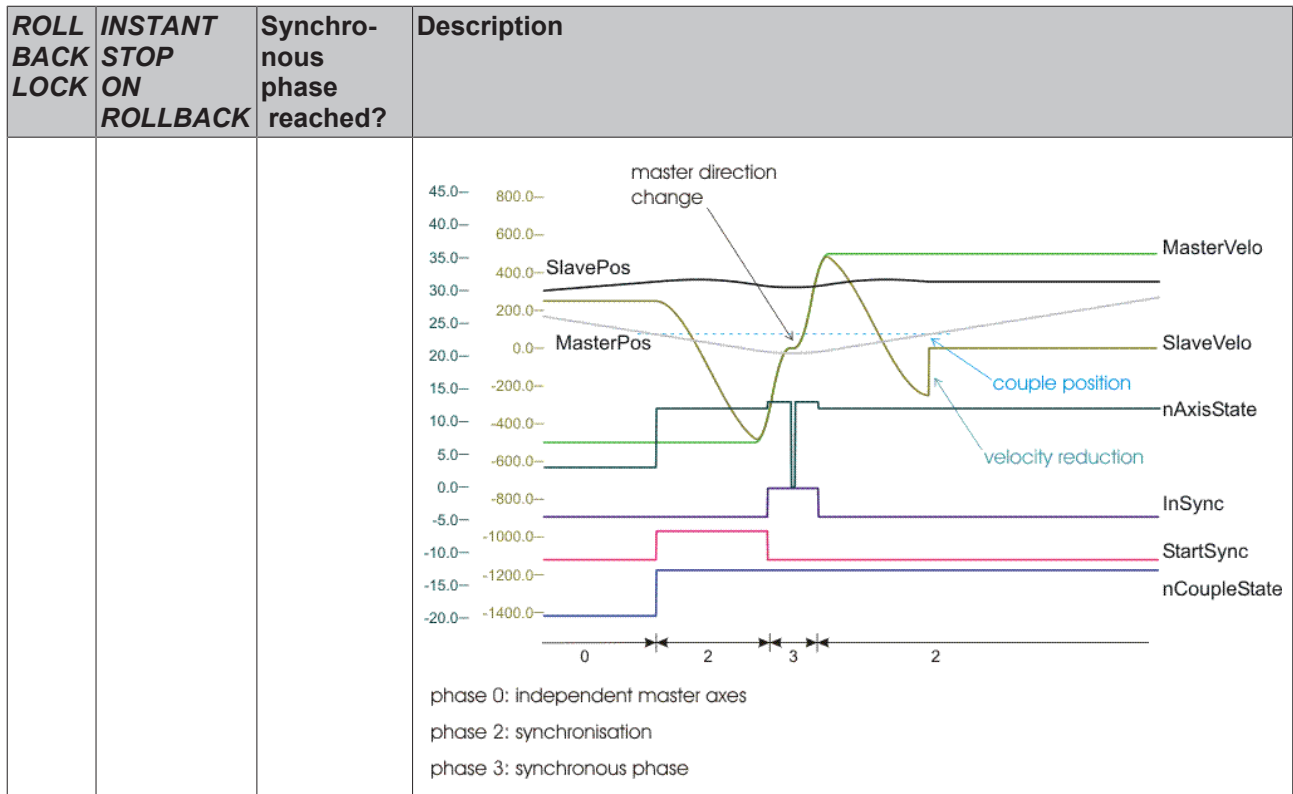
The effect of these two bits must be differentiated according to whether the synchronous phase is reached before the motion reversal, or whether the motion reversal already occurs in the synchronization phase.

The following overview explains in detail the effect of the *GEARINSYNC_OPMASK_ROLLBACKLOCK* and *GEARINSYNC_OPMASK_INSTANTSTOPONROLLBACK* bits.

ROLL BACK LOCK	INSTANT STOP ON ROLLBACK	Synchronous phase reached?	Description
0	0	yes	<p>Case 1:</p> <p>With the bit combination specified on the left, synchronous coupling is maintained for all master movements once the Universal Flying Saw is in the synchronous phase. In the figure below, a motion reversal of the master axis occurs in the synchronous phase, so that it moves backwards beyond the coupling position. The synchronous coupling is maintained here, so that the slave also moves backwards beyond the coupling position.</p> <p style="font-size: small;"> 50.0- 1000.0- 45.0- 800.0- change 40.0- 600.0- 35.0- 400.0- 30.0- 200.0- 25.0- 0.0- 20.0- -200.0- 15.0- -400.0- 10.0- -600.0- 5.0- -800.0- 0.0- -1000.0- -5.0- -1200.0- -10.0- -1400.0- -15.0- -20.0- </p> <p style="font-size: small;"> master direction SlavePos MasterPos SlaveVelo MasterVelo nAxisState InSync StartSync nCoupleState couple position 12 13 3 0 0 2 3 phase 0: independent master axes phase 2: synchronisation phase 3: synchronous phase </p>
0	0	no	<p>Case 2:</p> <p>Before reaching the synchronous phase, a motion reversal of the master axis takes place in the synchronization phase so that it moves backwards beyond the coupling position. Since the synchronous phase has not been reached, in this case the velocity of the slave axis is reduced with a 5th order polynomial when the coupling position is reached and the axis is brought to a standstill.</p>

ROLL BACK LOCK	INSTANT STOP ON ROLLBACK	Synchronous phase reached?	Description
			 <p>phase 0: independent master axes Phase 1: pre phase Phase 2: synchronisation</p>
0	1	no	<p>Case 3: As in case 2, except that the velocity of the slave axis is reduced to zero within one tick after reaching the coupling position. The velocity reduction in one tick can trigger the following error monitoring of the axis.</p>  <p>phase 0: independent master axes phase 2: synchronisation</p>
0	1	yes	<p>Case 4: The behavior is identical to case 1!</p>
1	0	no	<p>Case 5: The behavior is identical to case 2!</p>
1	0	yes	<p>Case 6:</p>

ROLL BACK LOCK	INSTANT STOP ON ROLLBACK	Synchron- ous phase reached?	Description
			<p>The bit combination indicated on the left is used to activate the backstop of the Universal Flying Saw. In the figure below, a motion reversal of the master axis occurs in the synchronous phase, so that it moves backwards beyond the coupling position. With this backward movement of the master axis, the velocity of the slave axis is reduced to zero with a 5th order polynomial as soon as the coupling position is reached. Reverse movement of the slave axis is therefore prevented in that the slave velocity is continuously reduced as soon as the coupling position is reached.</p>  <p>phase 0: independent master axes phase 1: pre phase phase 2: synchronisation phase 3: synchronous phase</p>
1	1	no	<p>Case 7: The behavior is identical to case 3.</p>
1	1	yes	<p>Case 8: The bit combination indicated on the left is used to activate the backstop of the Universal Flying Saw. In the figure below, a motion reversal of the master axis occurs in the synchronous phase, so that it moves backwards beyond the coupling position. During this backward movement of the master axis, the velocity of the slave axis is reduced to zero in one tick as soon as the coupling position is reached. Reverse movement of the slave axis is therefore prevented in that the slave velocity is reduced as soon as the coupling position is reached.</p> <p>The velocity reduction in one tick can trigger the following error monitoring of the axis.</p>



9 Diagonal saw

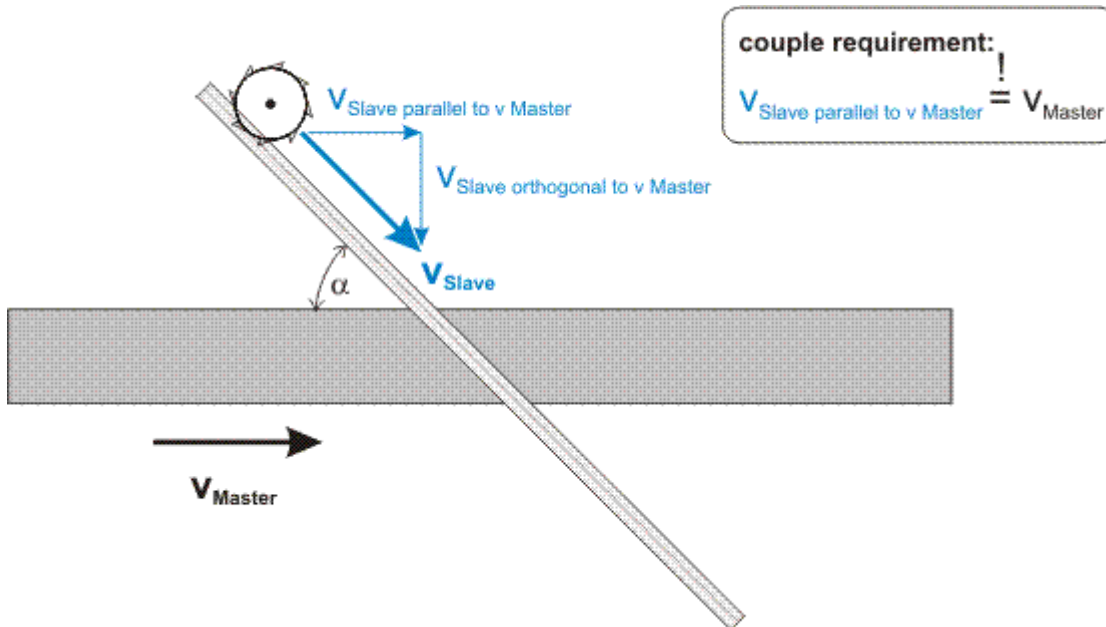
The simplest case of the diagonal saw involves synchronization to velocity.

In contrast to the Flying Saw, where the angle between the directions of the master and the slave movements is given, the coupling factor is given for the Universal Flying Saw. This coupling factor must be calculated in such a way that in the synchronized phase the component of the velocity of the slave axis in the direction of the master axis is the same as the master velocity. This means that the following equation is used to calculate the coupling factor:

$$F_{CouplingFactor} = \frac{1}{\cos(\alpha)} \cdot F_{GearRatio}$$

The angle α can not, of course, be 90° , because the slave axis would not then have any component of velocity in the direction of the master axis.

Schematic diagram of a "flying diagonal saw"



10 Interfaces

Cyclic axis interface

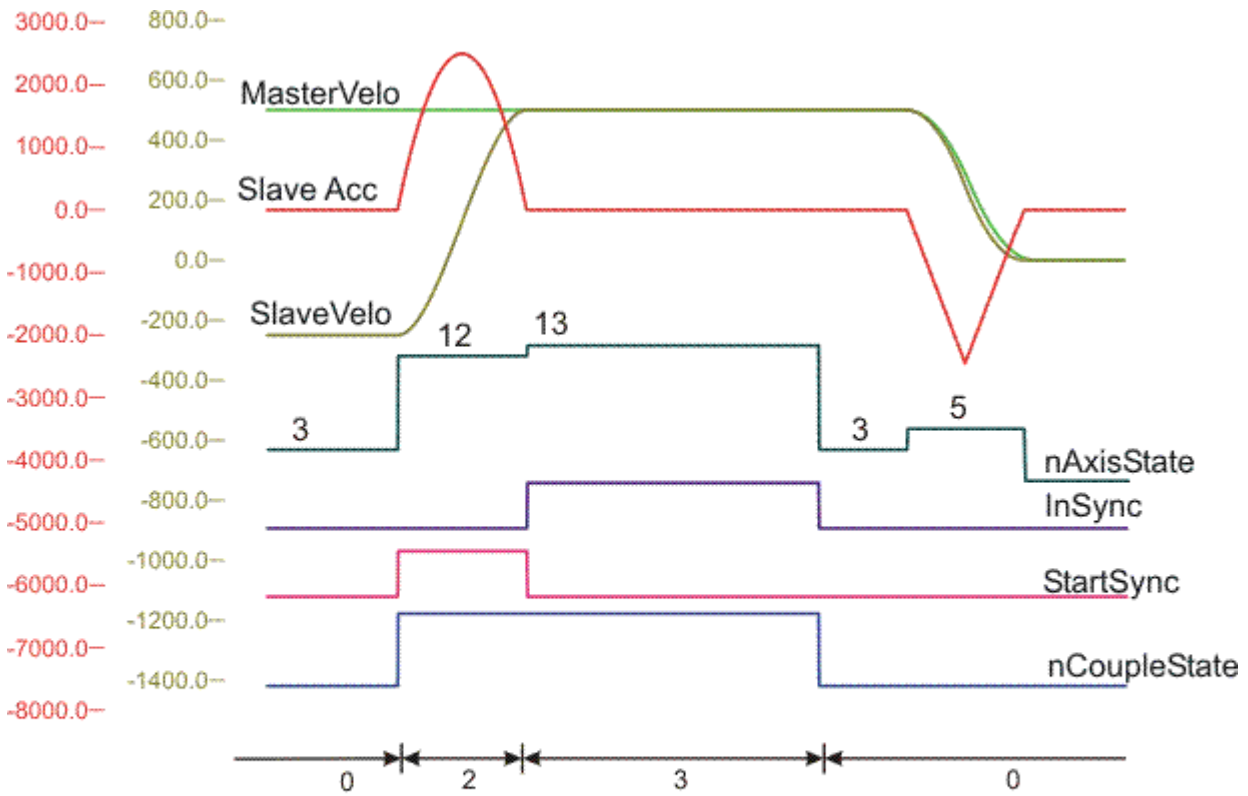
The variable `nAxisState` in the cyclic axis interface indicates the state of movement of the slave or master axis. The movement states of the slave axis are described in the table. All those states that are not listed are master states, and these are explained elsewhere.



For the time being the slave states only apply to slaves of the "Universal Flying Saw" type!

Present state of the axis movement / movement phases of the continuous slave axis (servo)			
nAxisState	Type	Description	
0	General	Generator not active	(INACTIVE)
other	Master-State	Explanation, see TwinCAT NC General documentation	
11	Slave-State	The slave is in a preliminary movement phase	(PREPHASE)
12	Slave-State	The slave is in the synchronization phase	(SYNCHRONIZING)
13	Slave-State	Synchronization of the slave has been achieved, and movement is synchronous	(SYNCHRON)

The diagram illustrates a typical coupling with the Universal Flying Saw. Before the Universal Flying Saw is started, there are two independent master axes (`nAxisState` < 10). The `nAxisState` = 12 after coupling has started in the synchronization phase. Achievement of the synchronized phase is indicated by `nAxisState` = 13. After the slave axis has been uncoupled from the master axis `nAxisState` is then < 10, which means that its movement state is again that of a master axis.



phase 0: independent master axes

phase 2: synchronisation

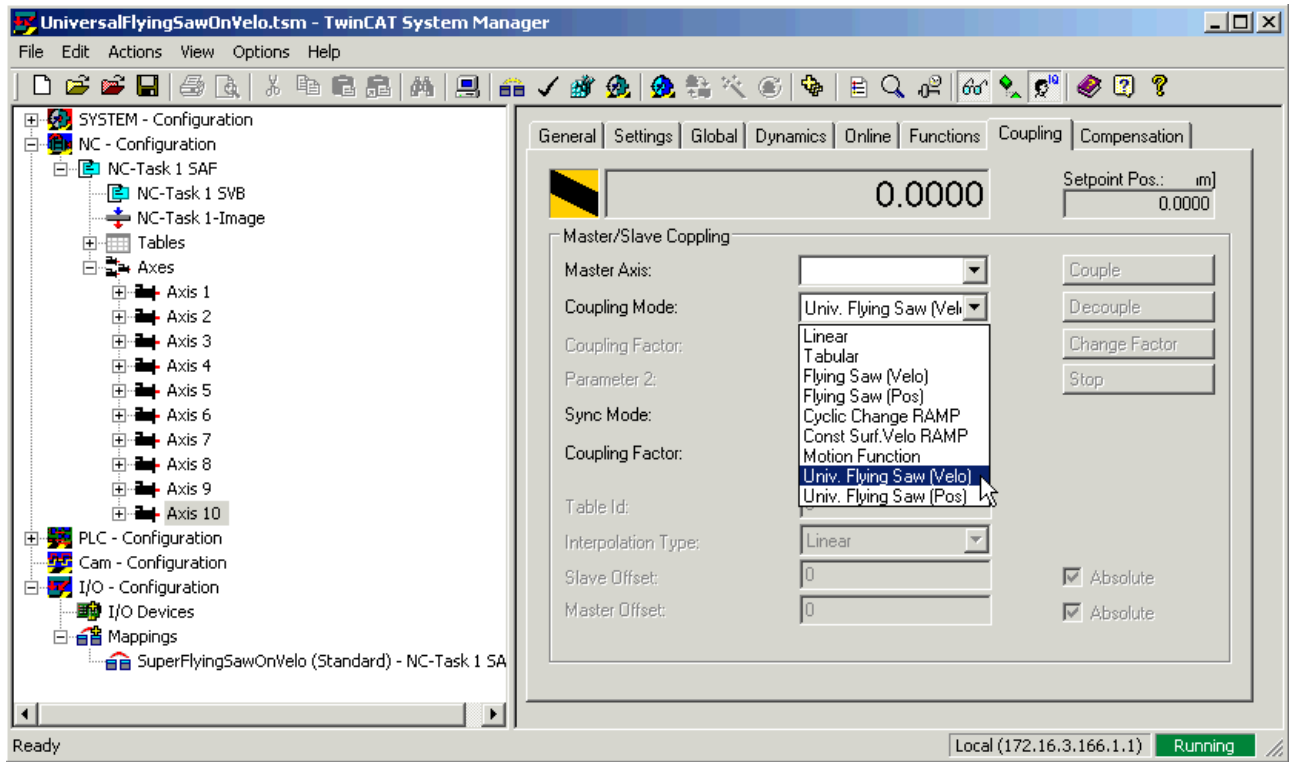
phase 3: synchronous phase

ADS interface

Documentation of the [ADS interface](#).

11 Operation from the System Manager

For commissioning purposes the Universal Flying Saw can also be started directly from the System Manager.



12 PLC API

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 **TcMC2_FlyingSaw.lib**, available as an *additional product*, provides easy management of the flying saw. An [example program using the flying saw \[▶ 44\]](#) makes use of this library.

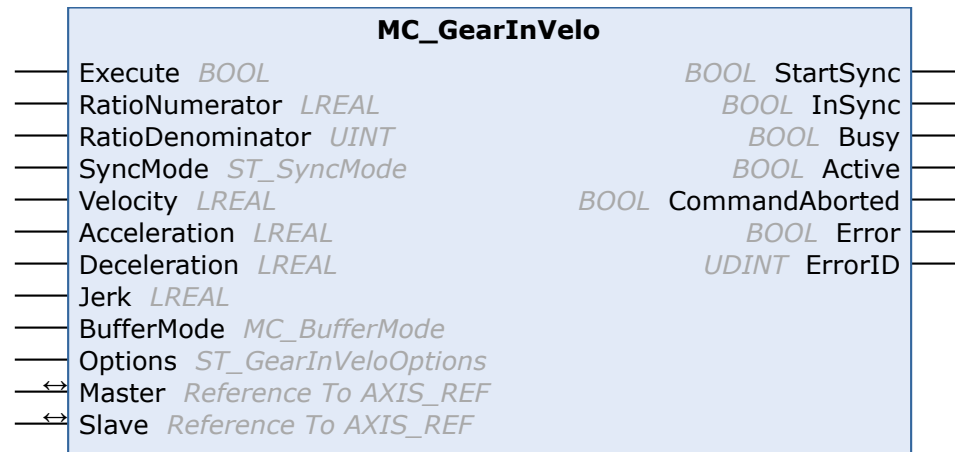
Zur einfachen Handhabung der Fliegenden Säge dient die TwinCAT PLC Library **TcMC2_FlyingSaw.lib**, die als *zusätzliches Produkt* erhältlich ist. Ein [Beispielprogramm zum Thema Fliegende Säge \[▶ 44\]](#) verwendet diese Bibliothek.

The following PLC function blocks are available for operation of the Universal Flying Saw.

Function	Block
Synchronisation to velocity	MC_GearInVelo [▶ 33]
Synchronisation to position	MC_GearInPos [▶ 36]
Read the characteristic values	MC_ReadFlyingSawCharacteristics [▶ 39]
Uncouple the slave	MC_GearOut [▶ 40]

12.1 TcMC2_FlyingSaw

12.1.1 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 <i>Execute</i> input.
RatioNumerator	Gear ratio numerator. Alternatively, the gear ratio can be specified as a floating point value, if the denominator is 1.
RatioDenominator	Gear ratio denominator
SyncMode	In the data structure <i>SyncMode</i> [▶ 41] 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. The velocity given here is only checked if this checking is activated through the <i>SyncMode</i> [▶ 41] 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. The acceleration given here is only checked if this checking is activated through the <i>SyncMode</i> [▶ 41] 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. The deceleration given here is only checked if this checking is activated through the <i>SyncMode</i> [▶ 41] 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. The jerk given here is only checked if this checking is activated through the <i>SyncMode</i> [▶ 41] 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 <i>Busy</i> output becomes TRUE when the command is started with <i>Execute</i> and remains TRUE as long as the command is processed. When <i>Busy</i> becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, <i>InSync</i> , <i>CommandAborted</i> or <i>Error</i> , 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, see Overview of NC errors (TC2) .

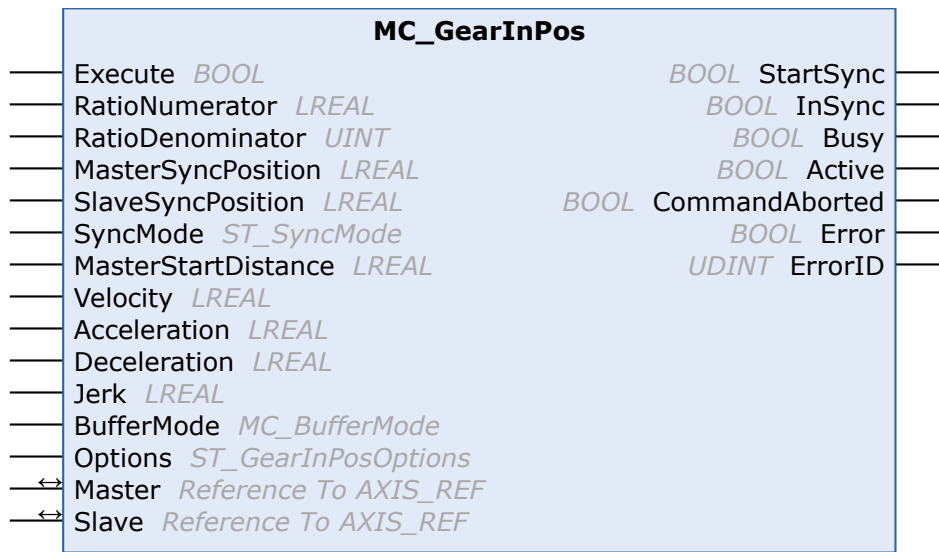
Inputs/outputs

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

Master	Master axis data structure.
Slave	Slave axis data structure.

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.

12.1.2 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 <i>Execute</i> input.
RatioNumerator	Gear ratio numerator. Alternatively, the gear ratio can be specified as a floating point 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 <i>SyncMode</i> [▶ 41] 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. The velocity given here is only checked if this checking is activated through the <i>SyncMode</i> [▶ 41] 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. The acceleration given here is only checked if this checking is activated through the SyncMode [► 41] 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. The deceleration given here is only checked if this checking is activated through the SyncMode [► 41] 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. The jerk given here is only checked if this checking is activated through the SyncMode [► 41] 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 <i>Busy</i> output becomes TRUE when the command is started with <i>Execute</i> and remains TRUE as long as the command is processed. When <i>Busy</i> becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, <i>InSync</i> , <i>CommandAborted</i> or <i>Error</i> , 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, see Overview of NC errors (TC2) .

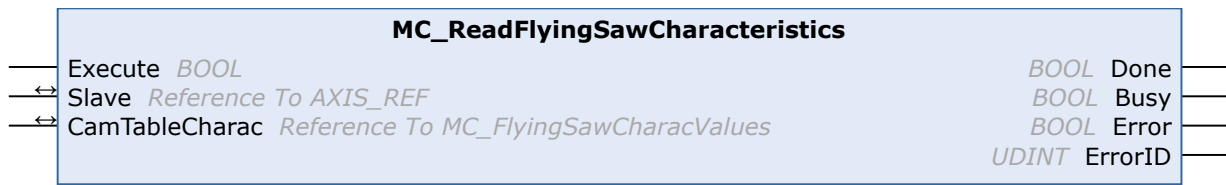
Inputs/outputs

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

Master	Axis data structure of the master.
Slave	Axis data structure of the Slave.

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

12.1.3 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.
----------------	--



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 <i>Busy</i> output becomes TRUE when the command is started with <i>Execute</i> and remains TRUE if the command is processed. When <i>Busy</i> becomes FALSE again, the function block is ready for a new command. At the same time one of the outputs, <i>Done</i> or <i>Error</i> , is set.
Error	Becomes TRUE, as soon as an error occurs.
ErrorID	Supplies the error number, see Overview of NC errors (TC2) .

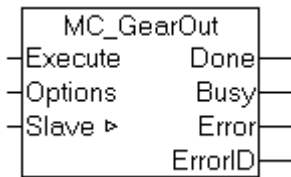
Inputs/outputs

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

Slave	Axis structure of the slave.
CamTableCharac:	Structure containing the characteristic values [► 42].

12.2 TcMC2

12.2.1 MC_GearOut



The function block *MC_GearOut* deactivates a master-slave coupling.

i If a slave axis is uncoupled during the movement, it is not automatically stopped, but reaches a continuous velocity with which it will continue to travel endlessly. The axis can be stopped with a *MC_Halt* or *MC_Stop*.

NOTE

If the setpoint generator type of the axis is set to "7 phases (optimized)", the slave axis assumes an acceleration-free state after uncoupling and continues to move with the resulting constant velocity. There is no positioning based on the master travel path calculated with the coupling factor. Instead, the behavior matches the behavior after a *MC_MoveVelocity* command. In TwinCAT 2.10, the setpoint generator type can be selected by the user. From TwinCAT 2.11, the setpoint generator type is set to "7 phases (optimized)". The behavior described here is the result of a project update from TwinCAT 2.10 to TwinCAT 2.11. Depending on the circumstances, an update of existing applications to version 2.11 may necessitate an adaptation of the PLC program.

Inputs

```
VAR_INPUT
    Execute   : BOOL;
    Options   : ST_GearOutOptions;
END_VAR
```

Execute	The command is executed with a rising edge at input <i>Execute</i> .
Options	Currently not implemented

Outputs

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

Done	Becomes TRUE, if the axis was successfully uncoupled.
Busy	The <i>Busy</i> output becomes TRUE when the command is started with <i>Execute</i> and remains TRUE as long as the command is processed. If <i>Busy</i> becomes FALSE again, the function block is ready for a new job. At the same time one of the outputs, <i>Done</i> or <i>Error</i> , is set.
Error	Becomes TRUE if an error occurs.
ErrorID	If the error output is set, this parameter supplies the error number.

Inputs/outputs

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

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.

12.3 Data types

12.3.1 ST_SyncMode

```

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;
  GearInSync_OpMask_IgnoreSlaveAcc : BOOL;
  GearInSync_OpMask_DetailedErrorCodes : 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

```



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

See also:

[Operation of the individual bits \[► 18\]](#)

[Error Codes](#)

12.3.2 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.

12.4 Example program

12.4.1 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:

https://infosys.beckhoff.com/content/1033/TS5055_TCNC_FlyingSaw/Resources/437571723/.zip

13 Error situations and error codes:

The following situations and/or parameters are not permitted, and result in errors:

- The Universal Flying Saw cannot at present be coupled if the master velocity at the coupling time is zero, but the slave velocity is not zero. A combination of zero master velocity and zero slave velocity is, however, permitted.
- Gearing factor is zero (otherwise the gearing factor can adopt any value)
- The maximum velocity, acceleration, deceleration, or jerk are negative (reference is made here to the axis data (machine data) in the system manager, which is interpreted as a magnitude without arithmetic sign)
- The synchronization positions with reference to the start positions at the time of coupling for the master must always be geometrically in the future, and
 - a.) must also be geometrically in the future for the slave if the coupling factor is greater than zero,
 - b) must be geometrically in the past for the slave if the coupling factor is less than zero.

NOTE

Using Motion function tables

A **motion function table** is employed internally to implement the function of the Universal Flying Saw. This internally used table must not be occupied by other types of table function (such as by a cam plate). Otherwise, the coupling will be rejected with an error code.

The table ID of the internally used table can be calculated through the following equation:

$$TableId_{FlyingSaw} = 256 - SlaveAxisId$$

Description of the error codes

Description of the error codes

More Information:
www.beckhoff.com/ts5055

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com

