

SIEMENS

Ingenuity for life

Industry Online Support

Home

SIMOTION Traversing Drive

Manual V4.0.8

<https://support.industry.siemens.com/cs/ww/en/view/36037374>

Siemens
Industry
Online
Support



Legal information

Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place. For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed at: <http://www.siemens.com/industrialsecurity>.

Table of contents

Legal information	2
1 Traverser – overview	5
1.1 SIMOTION Converting Toolbox	5
1.2 Uses and benefits of the standard application	5
1.3 Core functions	7
1.4 Secondary conditions of the application.....	8
2 Traverser functions	9
2.1 Definitions.....	9
2.2 Traversing mode	10
2.2.1 Continuously.....	10
2.2.2 Step wind.....	13
2.3 Spikes.....	15
2.4 Stroke	18
2.5 Conical coil profiles	20
2.6 Diameter calculation.....	21
3 Integration	22
3.1 Required Technology Objects.....	22
3.2 Interconnection between technology objects	22
3.3 Integration of the library.....	24
3.4 Necessary units and variables	26
3.5 Integration of the application	27
4 Project example	30
4.1 Description of the application example	30
4.2 Description of the HMI.....	34
4.3 Expansion to include additional traversing winder	39
5 Functional description	40
5.1 Data types	40
5.2 Enumeration types	41
5.3 Array data types	45
5.4 Data structures	46
5.5 FBTravCntrl.....	58
5.5.1 Input and output parameters	59
5.5.2 Functionality	61
5.5.3 Error Messages	65
5.6 FBCamCalc.....	67
5.6.1 Input and Output Parameters.....	67
5.6.2 Functionality	68
5.6.3 Error Messages	70
5.7 FCChkCalc.....	71
5.7.1 Input and Output Parameters.....	71
5.7.2 Functionality	72
5.7.3 Error Messages	73
5.8 FBCalcPrior	75
5.8.1 Input and Output Parameters.....	75
5.8.2 Functionality	76
5.8.3 Error Messages	78
5.9 FBChkPrior.....	80
5.9.1 Input and Output Parameters.....	80
5.9.2 Functionality	82
5.9.3 Error Messages	82

Table of contents

6	FAQ	83
6.1	Layer dependant adaptations.....	83
6.2	Winder axis diameter calculation	83
6.3	Impact on winder speed	83
7	Collection of formulas.....	84
7.1	Coil profile definition formulas	84
7.2	Diameter calculation.....	84
7.3	External diameter rounding	85
7.4	Calculating the maximum length of the cam	86
7.5	Adapting the stroke and the spike in the additional velocity mode.....	86
7.6	Adaptation of the spike according the position offset mode.....	86
7.7	Calculating the motion profile	87
7.8	Abbreviations.....	90
8	Appendix	91
8.1	Service and Support.....	91
8.2	Application Support	92
8.3	Links and Literature	92
8.4	Change documentation	92

1 Traverser – overview

1.1 SIMOTION Converting Toolbox

The subsequently described library to implement a traverser application is included in the SIMOTION Converting Toolbox:

<https://support.industry.siemens.com/cs/ww/en/view/109744606>

The SIMOTION Converting Toolbox was developed with the objective of providing standard applications for typical converting machines.

Converting involves the processing of a material in a certain way. In most cases, this process starts with unwinding a material web - which is then processed in one or several steps. Typical machines include:

- Printing machines
- Post press machines (downstream of the actual print process)
- Coating machines
- Laminating machines
- Cutting machines
- Material web inspection machines
- Paper finishing machines
- Wire-drawing machines
- Textile machines
- Diaper making machines
- ...

The standard applications are based on open-source libraries. These include tested and documented blocks that support users when they develop, program and commission solutions. Structured block interfaces - as well as documented program code - allow adaptations to be made to address specific customer requirements.

1.2 Uses and benefits of the standard application

The standard SIMOTION traverser application is designed for machines where materials are traversed while they are being wound. Traversing means that the material is specifically positioned located on the coil or bobbin as it is being wound. Typical materials include wire, foil, textile fibers and cable.

The application has been developed with the goal of creating a flexible solution for traversing applications, where the relevant data that is important for the traversing process can be specified via input values - and can be changed in operation. This means that by specifically entering input values, the traversing profile - and therefore the stability of the coil/or bobbin being wound - can be specifically influenced. As a result of the openness of the software structure, when necessary, it is also possible to make changes to the function blocks provided in the library.

With the appropriate equipment, the application allows the widest range of materials to be traversed (e.g. textile fibers, wires, foils, cables,...). It is ensured that the motor, which is driving the traversing mechanism, follows the traversing

profile, defined by the input values, with the highest precision. In practice, the quality of traversing motion depends on the mechanical design of the traversing unit as well as on the attributes of the material to be traversed.

The traversing drive runs in synchronism with a winder, which provides the leading value for the traversing. The winder is typically implemented using the standard LConLib application.

<https://support.industry.siemens.com/cs/de/en/view/35818687>

1.3 Core functions

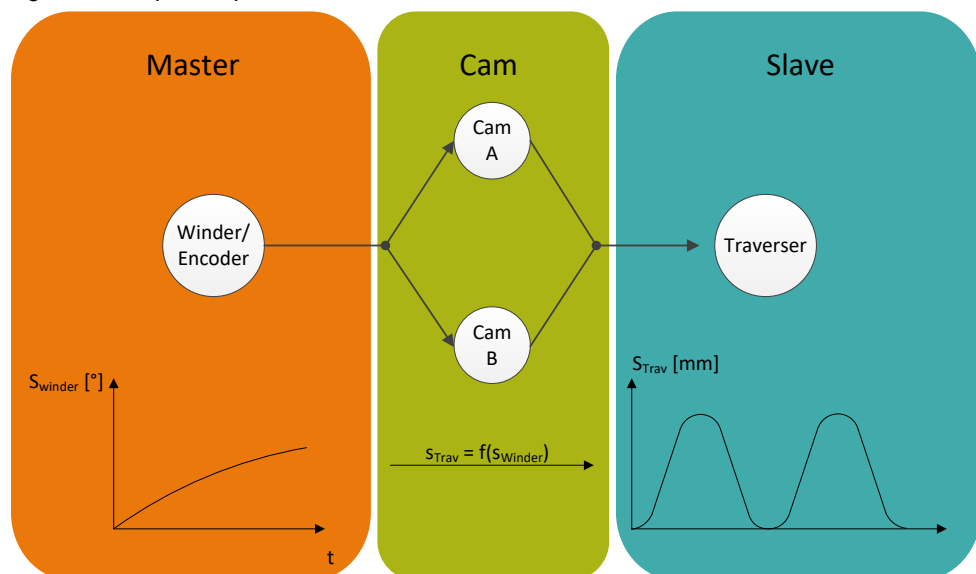
The standard traverser application calculates the motion profile and controls the axis in the automatic mode. Users must set the parameters before the program is activated. In the automatic mode, the traverser traverses in synchronism with the winder axis via a calculated cam relationship until the automatic mode is interrupted or a fault occurs.

In the automatic mode, user parameters are continually checked in order to identify any parameter changes. If a change was detected, then a new cam is automatically calculated and becomes active. In addition, the application includes the following functions:

- Cam change at the edge point or immediately
- Immediate reversal of the traverser
- Selection of different traversing modes (Continuously / Step wind)
- Automatic / manual mode

The principle of operation of the traverser application is shown in the following diagram:

Fig. 1-1 Principle of operation



For the manual mode, the application provides users with their own user interface. This can be used for setting-up (e.g. referencing/homing, jogging, positioning etc.) the traverser axis. This has been implemented in the project example attached.

1.4 Secondary conditions of the application

Technology objects

- Master value (winder):
 - External encoder, Positioning axis, Following axis
 - Rotary axis
- Traverser:
 - Following axis
 - Linear axis
 - For profile calculation additionally two cams must be added

2 Traverser functions

2.1 Definitions

Traverser

Typically, the traverser axis is a linear axis whose position is defined in mm. The traverser moves between the adjustable coil edge points (position A and position B).

Traversing cycle

From the perspective of the traverser, the left-hand end of coil point is defined as position A, and the right-hand end of coil point as position B. The value of position B is always greater than the value of position A. Position A represents the start and end of a complete traversing cycle.

Motion profile

The motion profile defines the synchronization of the absolute position of the traverser with the relative position (angular offset) of the coil during a traversing cycle. The traverser motion profile is essentially defined using the following four parameters:

- **Acceleration angle**
The acceleration angle defines the angle that the coil moves through while the traverser is accelerating/ decelerating.
- **Waiting angle**
In order to establish stable edge points of the coil being wound, the traverser dwells for a specific time at these edge points, while material is added to the coil. The duration of the waiting angle refers to the angle that the coil traverses through.
- **Winding step**
The winding step defines the stroke of the traverser in millimeters during one coil revolution. For values less than the material width, then the material is overlapped when winding, for example.
- **Displacement angle**
When calculating the profile, the duration of a traversing cycle is defined from these three parameters – acceleration angle, waiting angle and winding step. This describes how many degrees the coil moves through while the traverser moves from position A to position B - and returns.

At the instant that the traversing cycle starts, the coil is at an angle x . After the traversing cycle has been completed, when the traverser has returned to position A - and also the waiting angle at position A has expired - then the winder is at an angle y . The difference between angle x and angle y is called the displacement angle. In the most unfavorable scenario, the acceleration angle, winding step and waiting angle were selected so that a displacement angle of zero degrees is obtained. This would mean that the start angle and target angle are the same. Due to acceleration/deceleration as well as the waiting angle of the traverser, this results in material being accumulated at the coil at angle x . Entering a defined displacement angle would be the better solution. As a consequence, the material can be distributed at the edge points, and material does not accumulate at one position.

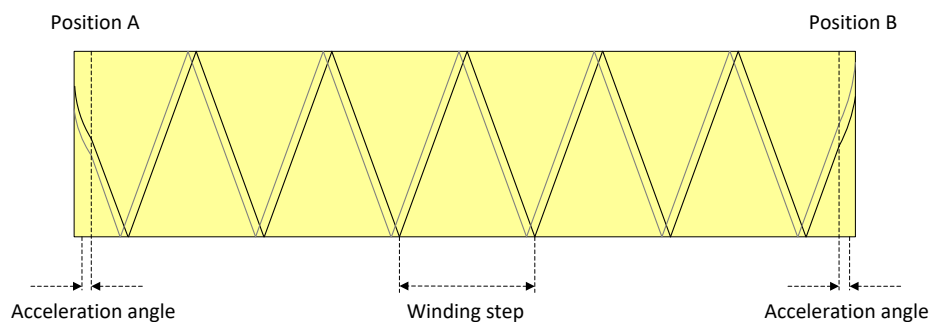
The angles defined in the parameters always refer to the angle of the coil shaft. In some instances, the precise meaning of the individual parameters depends on the traversing mode being used, and this will be described in more detail in the following chapter.

2.2 Traversing mode

2.2.1 Continuously

In this traversing mode, the traverser moves continuously between the adjustable coil edge points A and B. This is shown as example in [Fig. 2-1](#) for two traversing cycles.

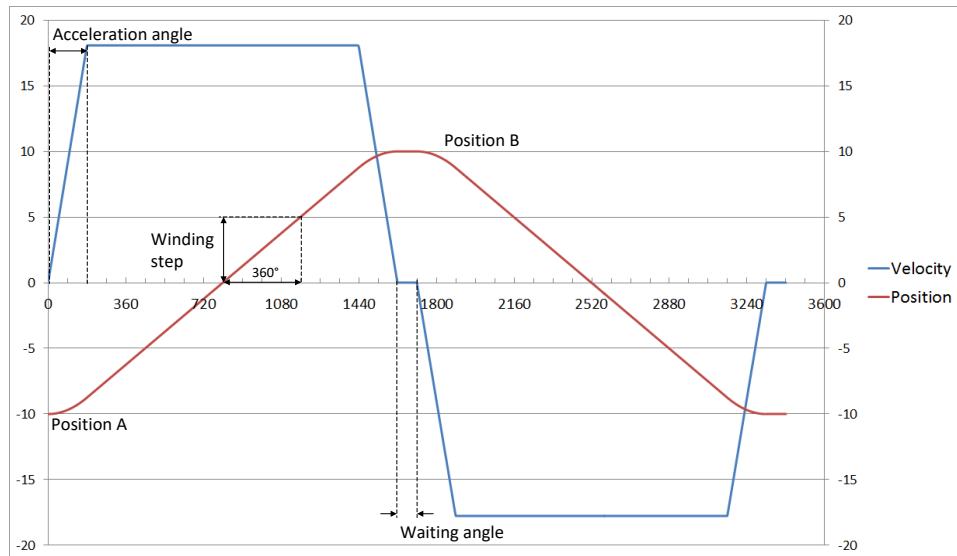
Fig. 2-1 Continuous traversing mode



To clearly show the significance of this parameter, the position/velocity profile for a continuous traversing cycle is shown in [Fig. 2-2](#). The X axis shows the position of the coil (angle) - the Y axis the position (mm) and velocity (mm/s) of the traverser. The following values were parameterized:

- Position A = -10 mm
- Position B = 10 mm
- Acceleration angle = 180°
- Waiting angle = 90°
- Winding step = 5 mm/rev
- Resulting displacement angle = $3420^\circ \text{ MOD } 360^\circ = 180^\circ$

Fig. 2-2 Position/velocity profile Continuously

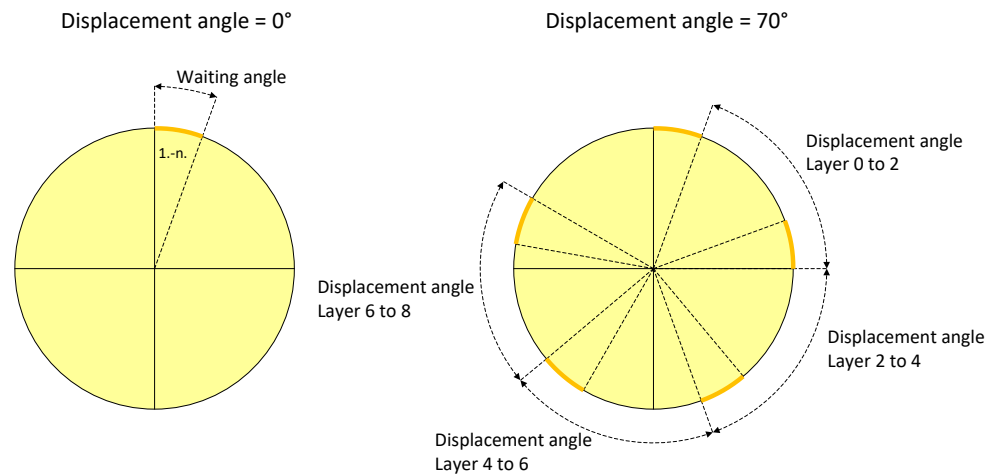


During the acceleration angle, the traverser is accelerated from standstill up to the constant velocity defined by the winding step - or is decelerated from the constant velocity down to standstill. At the two edge points, the traverser waits for the defined waiting angle before it starts in the opposite direction. While traveling at constant velocity, the traverser moves with a defined winding step of five millimeters per revolution of the coil.

Displacement angle

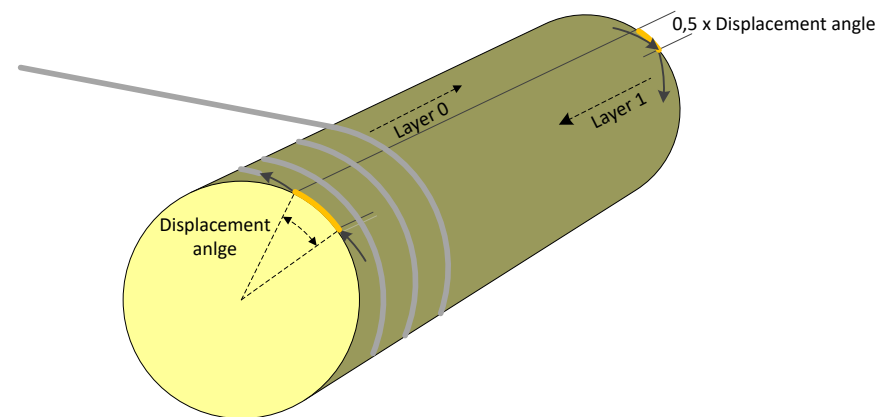
A defined displacement angle should be entered to avoid excess accumulation of material at the edge points. The side view of the coil at position A for two different parameterized displacement angles is shown in the following diagram:

Fig. 2-3 Influence of the displacement angle



For a resulting displacement angle of 0°, for example, a parameterized waiting angle would always be effective at the same coil position for all layers. By parameterizing a displacement angle, the material is distributed at the edge points around the complete circumference of the coil.

Fig. 2-4 Diagram showing the displacement angle

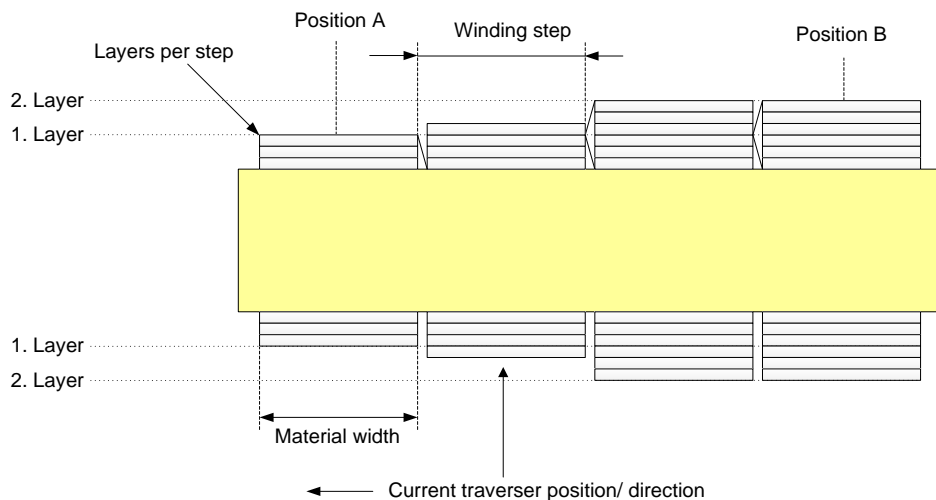


This means that when a defined displacement angle is reached, one of the three other parameters must be adapted within a defined interval. This is the reason that in the traverser application, for acceleration angle, waiting angle and winding step, one setpoint as well as one interval (minimum, maximum) can be defined for each of these values. Which parameter is adapted to maintain the displacement angle can be defined based on priorities. If it is not permissible to adapt a specific parameter, the corresponding priority must be set to zero. A detailed description of the calculation modes - as well as assigning priorities - can be taken from the function description of blocks [FBCamCalc](#) and [FBCalcPrior](#).

2.2.2 Step wind

In this particular mode, the traverser moves between the coil edge points; however, it dwells at certain positions, which means that the material is wound to create a sort of disk. After a specified number of layers have been wound on a disk, the traverser is moved through one winding step to the next position. The traversing cycle starts at position A; at this position, the traverser dwells until the number of layers has been wound on the coil. After the specified number of layers has been reached, the traverser moves through one winding step. The resulting position B is automatically calculated based on position A, the defined winding step and the number of steps. When reversing, at positions A and B twice the number of layers is wound before the traverser moves to the next position. The distance that the coil moves through while the traverser axis is being positioned is defined using the acceleration angle. Fig. 2-5 shows a traverser in the step wind traversing mode (reverse motion towards A in the second layer).

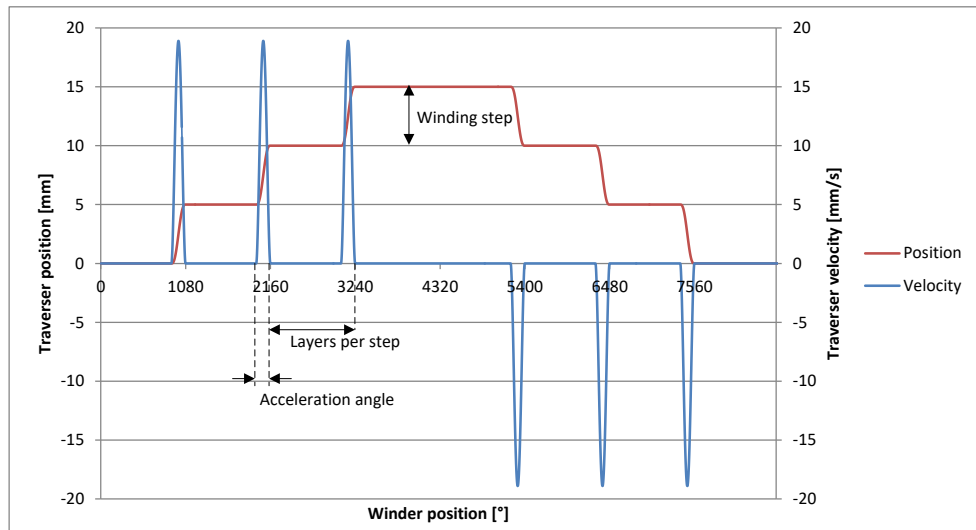
Fig. 2-5 Step wind traversing mode



The position/velocity diagram in the step wind mode is shown in the following diagram. The X axis shows the position of the coil (angle) - the Y axis the position (mm) and velocity (mm/s) of the traverser.

- Position A = 0 mm
- Resulting position B = 15 mm
- Number of layers per step = 3
- Waiting angle at the end of each step = 0 °
- Waiting angle at the edge points of the coil = 0°
- Acceleration angle = 180°
- Winding step = 5 mm/rev

Fig. 2-6 Position/velocity profile Step Wind



The diagram shows a traversing cycle in the Step Wind traversing mode. The cycle starts at position A, where the traverser dwells. The specified number of layers is wound at this position. Further, it is possible to parameterize a waiting angle at the end of each step. The arm then moves to the next position defined by the length of the acceleration angle. The cam profile is formed using a fifth order polynomial. The angle, referred to the coil axis - where the traverser dwells at a certain position - can be calculated as follows:

$$AnglePerStep = 360^\circ * r32LayerPerStep + waitingAngle - accelerationAngle$$

Using the waiting angle, the influence of decelerating and accelerating can be distributed over the circumference of the coil - similar to the displacement angle in the continuous traversing mode. Alternatively, this can also be achieved by having a non-integer number of layers per step. Further, it is possible to parameterize a separate waiting angle at the coil edge points A and B.

Note

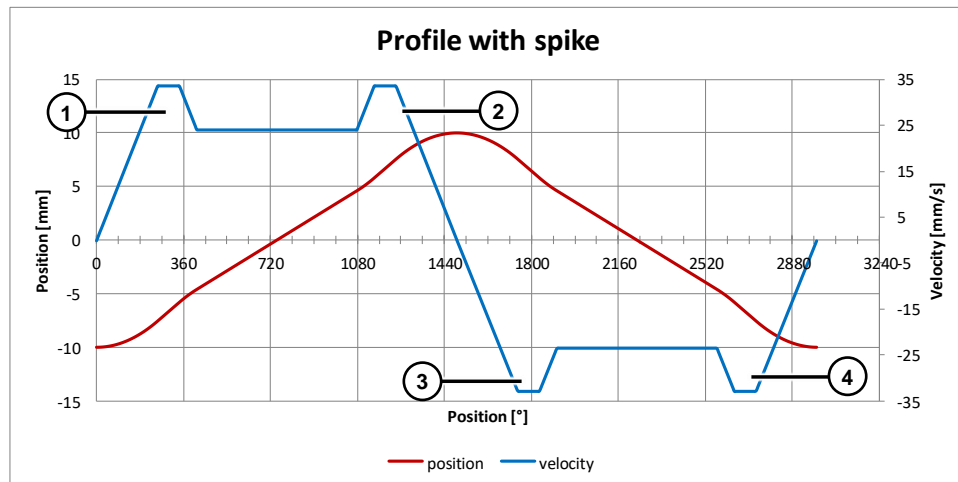
In the step wind mode it is not possible to configure a defined displacement angle. The displacement angle is obtained from the parameterization for:

- Winding step
- Number of steps
- Number of layers
- Acceleration angle
- Waiting angle (at the end of the step and at the edge points)

2.3 Spikes

The spike function is used to control the accumulation of material at the edge of the coil as a result of acceleration or deceleration losses. A spike can only be parameterized in the "Continuously" traversing mode. To implement this function, the motion profile is extended to include additional segments in which the traverser can be parameterized so that it moves faster than the velocity defined by the winding step.

Fig. 2-7 Spike configuration



Note Spike functionality is only available in the "Continuously traversing" mode

There are 2 modes for defining the spike:

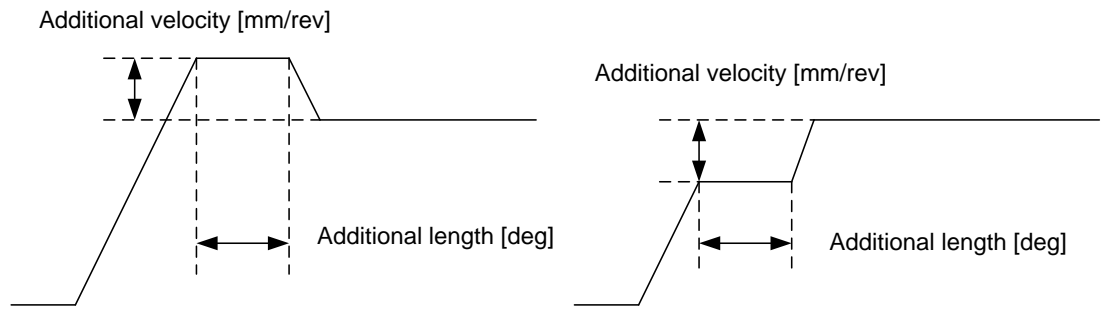
Definition with additional velocity

In this mode, a spike is defined by the spike length and the additional velocity (refer to Fig. 1-4).

The spike length is defined in degrees referred to the winding axis and defines the part of the spike in which the traverser moves with a constant velocity. The velocity offset is specified in mm/360° (the same as the winding step).

The acceleration and braking phases of the spike are not considered in this mode. The acceleration and the deceleration of the spike have the same absolute value as the acceleration or deceleration of the motion profile at the end points of the coil.

Figure 2-8 Spike mode – additional velocity

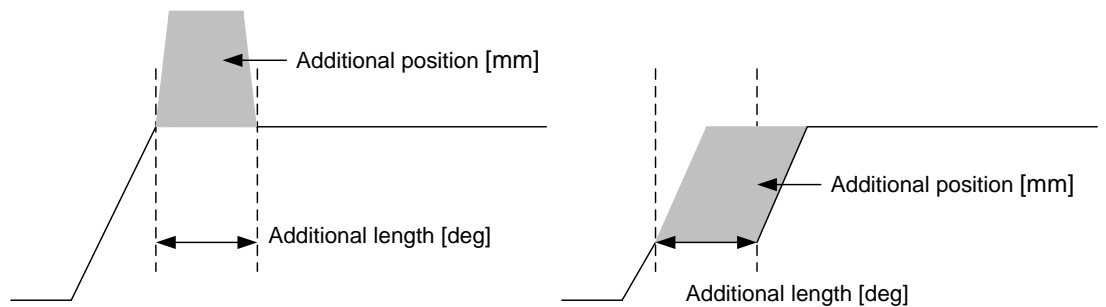


Definition with position offset

In this mode, the spike is defined by the spike length and the position offset of the traverser, as displayed in Figs. 1-5 and 1-6.

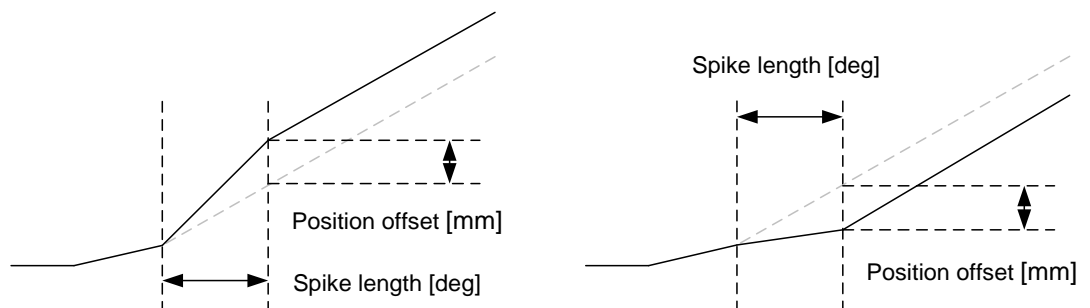
The velocity offset – including the acceleration and deceleration – are calculated from the specified parameters and cannot be influenced by the user. For the position offset, a linear interpolation is made between the parameterized points; acceleration and deceleration can, if necessary, be limited by the axis settings

Figure 2-9 Spike definition mode position offset – velocity



The surface under the trapezoidal shape corresponds to the position offset. When compared to the additional velocity mode, the position offset is directly parameterized in this mode.

Figure 2-10 Spike definition mode position offset – position



In the position offset mode all spikes must have the same sign. The spikes can still be activated/deactivated independently.

Adaptation of the spike

The following optional functions are available to adapt the spike parameter:

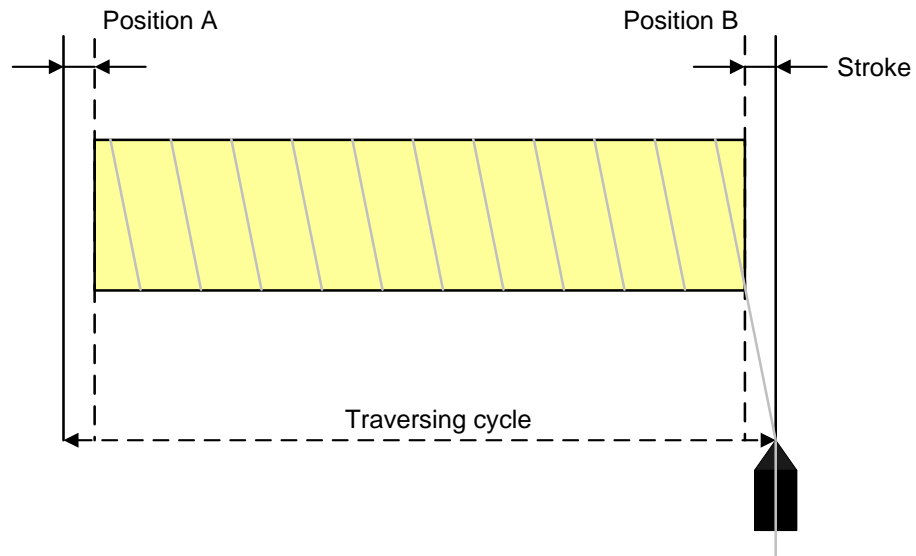
Definition with additional velocity:

- The velocity offset of the spike is adapted in the same ratio as the stroke
- Definition with position offset: The user can parameterize a correction value to reduce the absolute position offset per layer.

2.4 Stroke

The stroke function is used to ensure that the material reaches the end of coil points. This is important as the material cannot always be traversed perpendicular to the coil. To ensure that the material reaches the end of coil locations while traversing, the traverser must move beyond the defined end points - position A and B (see [Fig. 2-11](#)). The stroke function has the unit mm.

Fig. 2-11 Extension of the traverser path using the stroke function



Note

When activating the stroke function, the coil edge points are obtained according to the following formula:

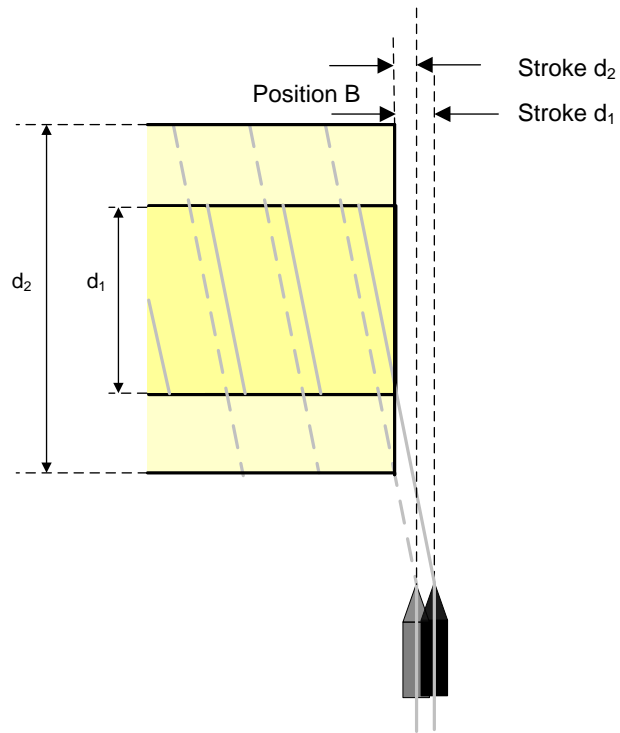
$$\text{Position A with stroke} = \text{position A} - \text{stroke}$$

$$\text{Position B with stroke} = \text{position B} + \text{stroke}$$

Adapting the stroke

The stroke depends on the distance between the coil surface and the traverser, meaning on the coil diameter. Optionally, the stroke can be automatically adapted depending on the ratio of the coil core to the coil diameter. This is shown as example in the following diagram for two coil diameters d_1 and d_2 .

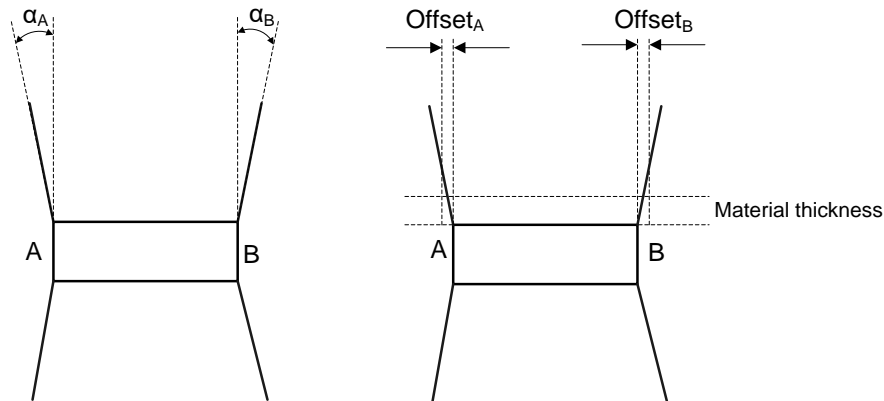
Fig. 2-12 Adaptation of the stroke function



2.5 Conical coil profiles

Depending on the particular field of application, many different coil types with different shapes are used in the industrial environment (see [Fig. 2-13](#)). When compared to cylindrical coils, for conical coils, the position A and position B change in each layer. As a consequence, a new cam must be calculated for each new layer. The following coil profiles can be configured based on an offset or angle at the edge of the coil:

Fig. 2-13 Coil profiles



Definition with the coil edge angle

In this mode, the coil profile is defined using the coil edge angle.

If 0° is parameterized, then the coil end locations are not adapted, and the coil width remains constant. With positive values, the coil width increases with increasing diameter - with negative angles, the coil width decreases.

For this mode, the precise coil diameter must be known (see chapter [Diameter](#).)

Definition layer offset

In this mode, the coil profile is adapted using a layer offset. The unit of this parameter is defined as μm (1/1000 mm).

With positive values, the coil width increases with an increasing number of layers - with negative values, the coil width decreases. This mode refers to the layer counter, and therefore does not require a precise diameter or material thickness.

Note

For conical coils, a new motion profile of the traverser must be calculated in each layer in order to maintain the required setpoints. The profile is calculated at position B, and becomes active at position A.

2.6 Diameter calculation

The diameter value can be used for the following functions:

- Adapting the stroke
With increasing diameter, the stroke value must become smaller (see [2.4](#))
- Conical coils defined using the coil edge angle
To define coil edges A and B for definition using an angle (see [2.5](#))
- To enter the diameter for a winder application

The traverser application supports two methods to determine the coil diameter based on the effective material thickness.

$$\text{Effective thickness} = \text{material thickness} * \frac{\text{Material width}}{\text{Winding step}}$$

Internal diameter calculation

The traverser application calculates the diameter, based on:

- The diameter of the coil core
- The effective material thickness
- The layer counter

$$\text{Diameter} = \text{coilcore} + \text{layer counter} * 2 * \text{effective thickness}$$

External diameter

The diameter can also be determined outside the traverser function. To guarantee the calculation of a deterministic traversing profile, the external diameter is smoothed for the profile calculation using the effective material thickness.

3 Integration

3.1 Required Technology Objects

The following technology objects are required for the *SIMOTION Traversing Drive* application:

Winder axis or machine encoder at the winder axis

This technology object is used to sense the winding angle and must therefore be a real or virtual axis with position interface (POSAXIS or FOLLOWINGAXIS) or an external encoder (EXTERNALENCODERTYPE). This object is not controlled from this application – the appropriate control must be programmed by the user for the specific machine using for example SIMOTION Axis Function Block. This object is used as leading value source for the camming (cam synchronous operation) of the traversing axis.

Traversing axis

The traversing axis must be set-up as synchronous axis (FOLLOWINGAXIS). The reason for this is that the traversing axis follows the winder axis through a cam relationship. This axis must be controlled in the user program using for example SIMOTION FB LineAxis or SIMOTION Axis Function Block as the traversing function does not activate the axis, but controls the synchronous operation functionality. Further, no commands may be issued that influence the motion behavior of the axis as long as the traverser function is active (*FBTravCntrl*). The only exception to this rule is Emergency Stop. The axis must be powered-up and stationary (zero speed) when activating the traverser function.

The synchronous object of the traverser axis must be parameterized for a cam relationship using a setpoint coupling with the winder axis. If a machine encoder is used, the actual value coupling type must be set with extrapolation. In addition, the required cams must be assigned to the synchronous relationship between the winder and traverser axes.

Cams

A cam defines the motion relationship between traverser axis and winder axis. A new cam is calculated and automatically selected if the motion profile of the traverser axis is changed when adapting user parameters or as a result of automatic adaptation. In this particular case, two cam objects are required.

3.2 Interconnection between technology objects

The technology objects used by this application must be interconnected with the synchronous object of the traverser axis. The interconnections are configured in SIMOTION Scout in the configuration dialog box of the synchronous object.

Coupling between the winder axis and the traverser axis

The synchronous object of the traverser axis must be interconnected with the leading value source of the winder axis (either with the axis object or with the object of the machine encoder). Ideally, the coupling type is set to *Setpoint coupling*.

Figure 3-1 Coupling between the traverser axis and winder axis

Following axis:

Interconnections to the master value interface:

	TO name	Coupling type	Device
<input checked="" type="checkbox"/>	Axis_RED_Winder	Setpoint	D435

If a machine encoder is used, then the coupling type must be set to *Actual value coupling with extrapolation*. The precise extrapolation time can be determined using the *Extrapolation times for PB drives* calculation tool. This is provided on the SIMOTION Scout DVD.

NOTICE **Winder in torque controlled mode**

If the winder is controlled in a torque controlled mode using an over controlled speed controller an actual value must be used.

Figure 3-2 Coupling between the traverser axis and machine encoder

Following axis:

Interconnections to the master value interface:

	TO name	Coupling type	Device
<input checked="" type="checkbox"/>	Axis_RED_Winder	Actual value with extrapolation	D435

Interconnecting the cams

The cams configured for this application must also be interconnected with the synchronous object of the traverser axis.

Figure 3-3 The cam with the synchronous object of the traverser axis

Interconnections with cams:

	TO name
<input checked="" type="checkbox"/>	Trav01CamA
<input checked="" type="checkbox"/>	Trav01CamB



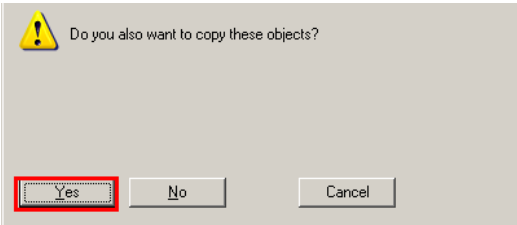
3.3 Integration of the library

The traverser functionality is part of the *LTravLib* library. This library is available as standard application *SIMOTION Traversing Drive* as well as through an XML Export. In order to use the functionality of the library, this must be integrated into the corresponding user project.

Copying from the application example

Two instances of SIMOTION SCOUT are opened to copy the *LTravLib* library. The application example is opened or dearchived in the first window and a new user program, which already includes the global libraries folder, is set-up in the second window.

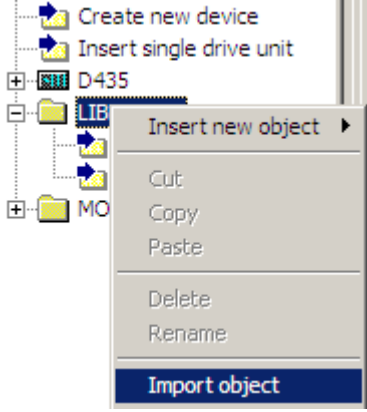
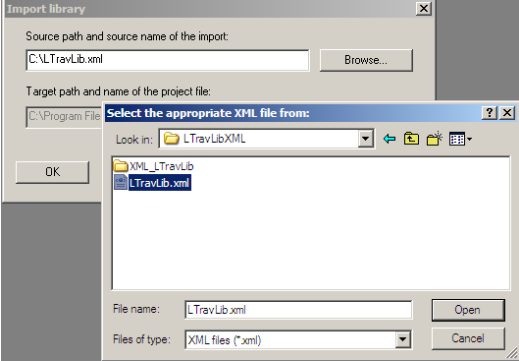
Table 3-1 Copying the library from the application example

Action	Comment
<p>The library is loaded into the buffer memory by right clicking on the library in the standard application and then Copy.</p>	
<p>The LTravLib library can be inserted if the Library folder is selected in the user project and then executed by clicking on it with the righthand mouse key and selecting paste.</p>	
<p>The lower-level objects are also copied.</p>	

Linking-in via XML import

With XML import, the *LTravLib* library is linked into the existing user project from an XML file.

Table 3-2 Importing the XML file

Action	Comment
<p>For the XML import, in the user project, the Library folder must be selected and then executed by clicking on it with the righthand mouse key. Using the Import object menu item, a window is now opened in which the path of the XML file must be specified.</p>	
<p>Using the Browse button, the path of the XML file is specified and the LTravLib library imported into the user project.</p>	

© Siemens AG 2020. All rights reserved.

Linking-in the library

To be able to use the library functions, types, function blocks etc. the library must be linked into the interface section of the corresponding unit.

```

INTERFACE
    USELIB LTravLib; // import library
END_INTERFACE
    
```

3.4 Necessary units and variables

For the basis functionality of the traverser, the two function blocks, *FBTravCntrl* and *FBCamCalc/FBCalcPrio* must be integrated into the user program. The cam calculation (*FBCamCalc/FBCalcPrio*) can only be executed in a motion task and *FBTravCntrl* only in a cyclic task. This means that global variables or structures must be defined to ensure the coordination and data transfer between the blocks.

Global variables are required for data exchange between the *FBTravCntrl* and *FBCamCalc/FBCalcPrio* function blocks. The required global variables are shown and described as an example corresponding to their function and interconnection in the following.

The prefixes stand for the following: g: global variable, bo: Boolean data type, s: data structure.

NOTE The function block *FBCalcPrio* supports the priority based profile calculation and uses internally the function block *FBCamCalc*. A detailed description of both function blocks can be found in the chapters [5.6](#) and [5.8](#). In case of layer dependant adaptations it is recommended to use the function block *FBCalcPrior* (see chapter [6.1](#)). The interconnection of both function blocks is identical so that in the following description only the function block *FBCalcPrio* is described.

Global structure instances

gsTravSetParamType	This structure includes user parameters for traverser operation. A detailed description is provided in the chapter for the type declarations
gsTravActParamType	Actual values – such as diameter, layer counter and profile-related data – are saved in this structure. A detailed description is provided in the Chapter for the type declarations.

Global control bits

gboStartCamCalc	Using this variable, the startCamCalc output of the <i>FBTravCntrl</i> block is transferred to the enable input of the <i>FBCalcPrio</i> block. This signals that a new cam calculation must be started.
gboCalcDone	This variable transfers the done output of the <i>FBCalcPrio</i> function block to the calcDone input of the <i>FBTravCntrl</i> block and therefore signals an error-free cam calculation.
gboCalcError	Using this variable, the error output of the <i>FBCalcPrio</i> is transferred to the calcError input of the <i>FBTravCntrl</i> in order to signal an error while the cam was being generated.
gboCalcBusy	This variable transfers the busy output of the <i>FBCalcPrio</i> to the calcBusy input of the <i>FBTravCntrl</i> in order to signal that the cam calculation is active.

3.5 Integration of the application

Calling the FBTravCntrl block in the user program

The *FBTravCntrl* block must be called in a cyclic task – general background in the user program. In order to do this, as already described, the block must be interconnected with the *FBCalcPrio* block using global variables.

The input parameters are explained in chapter [5.5.1](#).

Table 3-3 Call in LAD/ST

Parameters/variables	I/O symbols	Structures	Enumerations
	Name	Variable type	Data type
1	TravCntrl	VAR	fbtravcntrl (*LTravLib.lib)
2			

001 - Title	
	TravCntrl Fbtravcntrl 1
EN	ENO
Axis_RED_winder	startcamcalc
Axis_BLUE_Traverser	traverseraxis
Trav01CamA	travcama
Trav01CamB	travcamb
gstravsetparam	stravsetparam
hmi_inenabletrav	enable
hmi_inresetlayercount	resetlayercount
gboalcdone	calcdone
gboalerror	calcerror
gboalbusy	calcbusy
...	startcam
hmi_instartdir	startdir
hmi_reversenow	reversenow
hmi_windingmode	windingmode
gstravactparam	stravactparam


```

VAR
    TravCntrl : FBTravCntrl;
END_VAR
// Call FBTravCntrl

TravCntrl(winderAxis:= toMaster,
traverserAxis := Axis_BLUE_Traverser,
travCamA := Trav01CamA,
travCamB := Trav01CamB,
sTravSetParam := gsTravSetParam,
enable := HMI_InEnableTrav,
resetLayerCount := HMI_InResetLayerCount,
calcDone := gboCalcDone,
calcError := gboCalcError,
calcBusy := gboCalcBusy,
startCam := HMI_InStartCam,
startdir := HMI_InStartDir,
sTravActParam := gsTravActParam,
reverseNow := HMI_ReverseNow,
windingMode := HMI_WindingMode);

// Read FBTravCntrl outputs
gboCntrlBusy := TravCntrl.busy;
gboCntrlError := TravCntrl.error;
gboCntrlErrorId := TravCntrl.errorID;
gboStartCamCalc := TravCntrl.startCamCalc;
    
```

© Siemens AG 2020. All rights reserved

Calling the FBCalcPrio block in the user program

The *FBCalcPrio* block must be called in a motion task that is started with the rising and falling edge of the *startCamCalc* signal. The start of the motion task must be programmed in the user program.

Alternatively, the motion task used can be automatically started by the system if the SIMOTION control is brought into the RUN state. To do this, the corresponding motion task should be configured. In addition, the function block must be called in an endless loop.

Figure 3-4 Configuring Motion task for automatic start

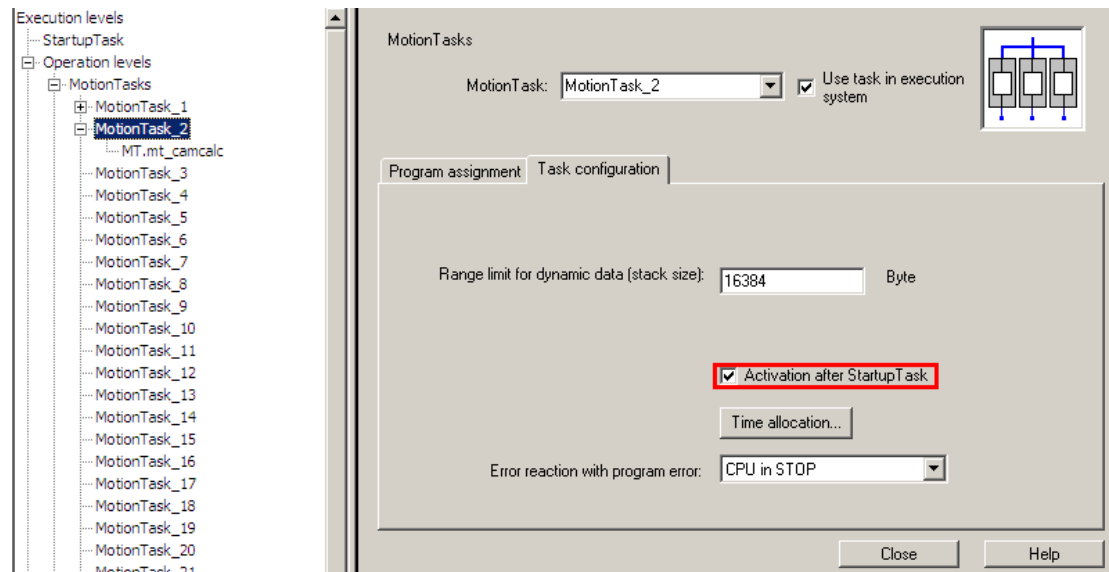


Figure 3-5 Call FBCalcPrio

```

// =====
PROGRAM pTravWinderMTCamCalc
//-----
//functionality:   In this program priority based profile calculation is called
//assignment:      Motion tasks
//-----

VAR
  aFBCalcPrior : ARRAY [0..NUM_OF_TRAVWINDERS-1] OF FBCalcPrior;
END_VAR

VAR_TEMP
  i32RetDint : DINT;
  u8Index : USINT;
END_VAR

WHILE TRUE DO

  FOR u8Index := 0 TO (NUM_OF_TRAVWINDERS-1) DO
    // Call CalcPrior
    aFBCalcPrior[u8Index](enable := gasTWTravCtrlOutput[u8Index].startCamCalc,
      sTravSetParam := gasTWTravSetParam[u8Index],
      sTravActParam := gasTWTravActParam[u8Index],
      ai8ModePriority := gai8TWModePriority[u8Index]);

    gasTWTravCtrlInput[u8Index].calcDone := aFBCalcPrior[u8Index].done;
    gasTWTravCtrlInput[u8Index].calcError := aFBCalcPrior[u8Index].error;
    gasTWTravCtrlInput[u8Index].calcBusy := aFBCalcPrior[u8Index].busy;
    gaboTWCalcErrorId[u8Index] := aFBCalcPrior[u8Index].ErrorId;

    IF gasTWTravCtrlOutput[u8Index].startCamCalc THEN
      gasTWActcalcMode[u8Index] := aFBCalcPrior[u8Index].eActcalcMode;
    END_IF;
  END_FOR;

  i32RetDint := _waittime(timevalue := T#0ms);

END_WHILE;
END_PROGRAM

```

Interconnecting function blocks FBTravCntrl and FBCalcPrio

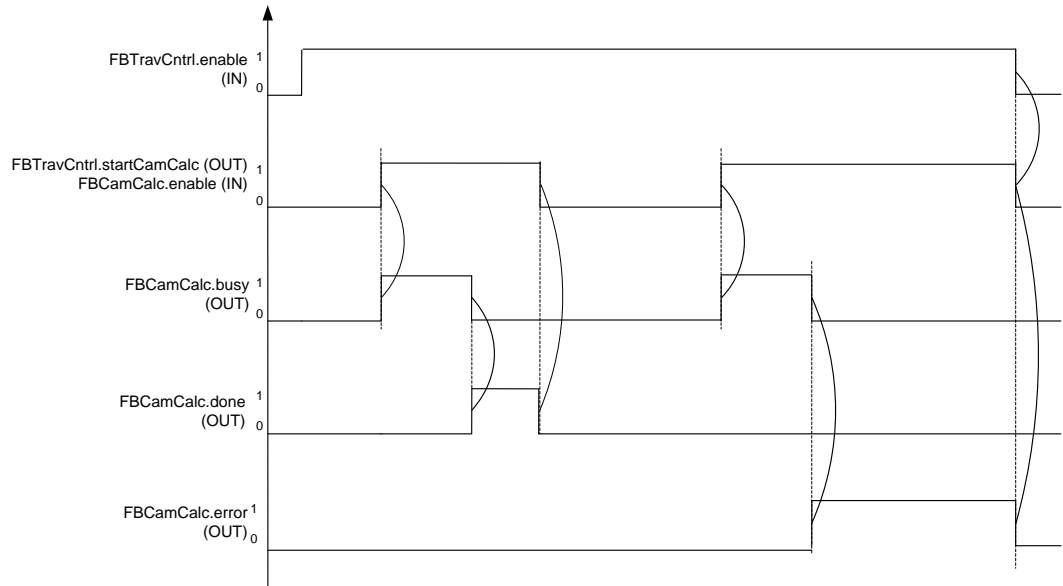
The *FBTravCntrl* and *FBCalcPrio* function blocks must be interconnected with one another precisely as described.

The *startCamCalc* output from *FBTravCntrl* must be interconnected with the *enable* input of the *FBCalcPrio*. This connection can either be established directly by connecting the blocks using global variables – or using the user program. After the function has started, the *busy* output of the *FBCalcPrio* switches to TRUE, until the new cam is calculated, the process is interrupted due to an error or the enable

input was again switched to FALSE (error = TRUE). The *busy*, *done* and *error* outputs of *FBCalcPrio* must be connected with the corresponding inputs of the *FBTravCntrl* (*calcBusy*, *calcDone*, *calcError*).

The switching sequence of the block inputs and outputs described is shown in the following signal diagram:

Figure 3-6 FB synchronization time diagram



Both blocks must access the same structure of the user parameters (*sTravSetParam*), i.e. the *sTravActParam* input of both blocks must be connected with the same structure instance. Otherwise, it cannot be guaranteed that both blocks will be synchronized.

Information regarding the user program

The user is responsible for integrating the function blocks into his automation solution or his user program. This includes, among other things, activating, referencing, positioning and jogging the axes, the operating states of the machine, the data management and alarm handling. Also included are the emergency shutdown and where relevant, the implementation of monitoring functions.

4 Project example

4.1 Description of the application example

In addition to the library, a project example based on a SIMOTION D435-2 is provided. The project is intended to explain the interaction between the individual technological components (traverser, winder, master axis), and includes the following components:

- Library LConLib (SIOS-ID: [35818687](#)) to control the winder axis
- Library LTravLib (SIOS-ID: [36037974](#)) to control the traverser axis
- Library LMCBasic (SIOS-ID: [48816191](#)) to control the virtual leading axis
- Sequence control
- HMI control

All programs, as well as the HMI, are also included in the library in folder "LTrav_ExampleBlocks" and "LTrav_ExampleHMI". The program blocks used and the technology objects required are listed in the following table.

Table 4-1 Expanding and adapting the application example

Program blocks used	Technology objects used
<ul style="list-style-type: none"> [-] D4x5 [D435-2 DP/PN] <ul style="list-style-type: none"> [+] X142 inputs/outputs [+] EXECUTION SYSTEM [+] ADDRESS LIST [+] GLOBAL DEVICE VARIABLES [+] AXES [+] EXTERNAL ENCODERS [+] PATH OBJECTS [+] CAMS [+] TECHNOLOGY [+] PROGRAMS <ul style="list-style-type: none"> [+] Insert ST source file [+] Insert MCC unit [+] Insert LAD/FBD unit [+] dConITDiag [+] dTravWinderHMI [+] pFault [+] pLine [+] pMainSequence [+] pTravWinder [+] pTravWinderHMI [+] SINAMICS_Integrated [S120 SINAMICS Integrated] [+] SCRIPTS [+] LIBRARIES <ul style="list-style-type: none"> [+] Insert library [+] LConLib [+] LMCBasic [+] LTravLib [+] MONITOR 	<ul style="list-style-type: none"> [-] D4x5 [D435-2 DP/PN] <ul style="list-style-type: none"> [+] X142 inputs/outputs [+] EXECUTION SYSTEM [+] ADDRESS LIST [+] GLOBAL DEVICE VARIABLES [+] AXES <ul style="list-style-type: none"> [+] Insert axis <ul style="list-style-type: none"> [+] axLine01 [+] TW01axTraverser [+] TW01axWinder [+] EXTERNAL ENCODERS [+] PATH OBJECTS [+] CAMS <ul style="list-style-type: none"> [+] Insert cam with CamEdit [+] Insert Cam with CamTool [+] CamATW01axTraverser [+] CamBTW01axTraverser

Technology objects

The following technology objects are required for the traverser application:

- Traverser axis as synchronous axis ("TW01axTraverser")
- Three cams to calculate the profile ("CamATW01axTraverser", "CamATW02axTraverser")
- Winder as positioning axis ("TW01axWinder")
- Virtual leading axis as positioning axis ("axLine01")

Program blocks

The project example shows the implementation of a traversing winder in conjunction with a virtual leading axis as machine master. The following standard libraries are used to implement the individual technological components:

- Control of the winder in the closed-loop control mode "V-Constant" using function blocks from the LConLib library
- Traverser control using function blocks from the LTravLib library
- Control of the leading axis using function blocks from the LMCABasic library

Fig. 4-1 Libraries that are used

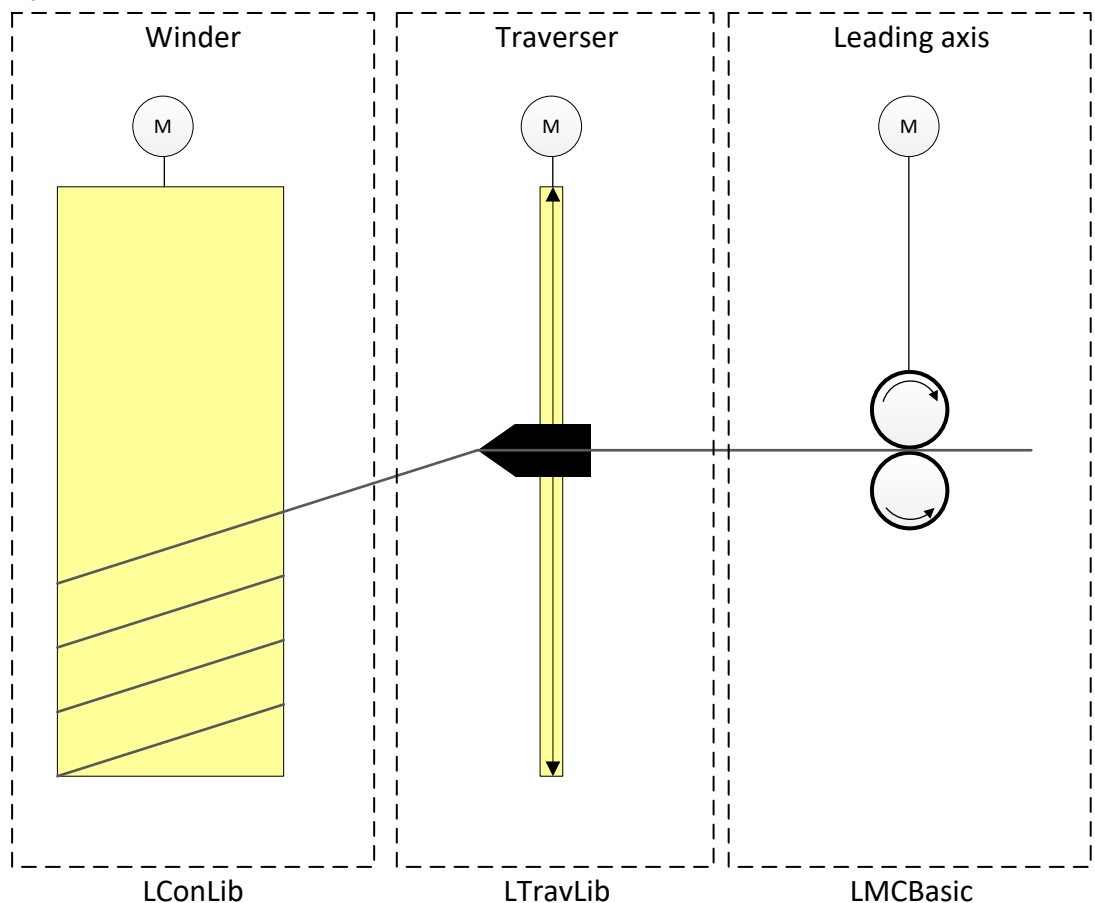


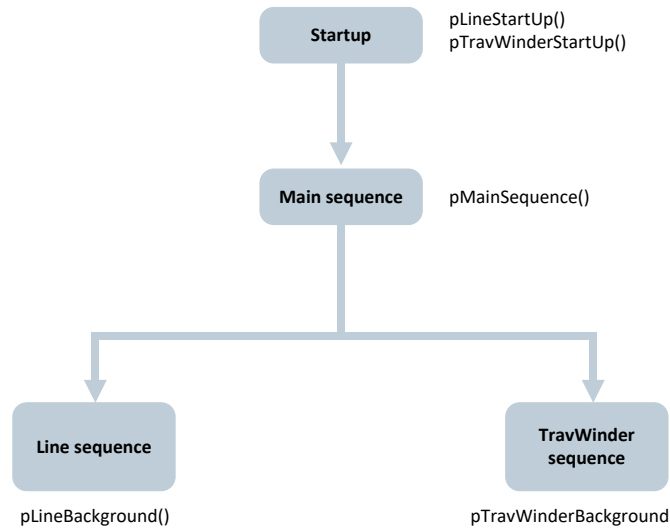
Table 4-2 Description of the program units

Unit	Description
dConITDiag	Contains constants and data types for using the IT-Diag web pages of the SIMOTION webserver
dTravWinderHMI	Definition of global data structures for the HMI
pFault	Example programs for the TechnologicalFaultTask and PeripheralFaultTaks
pLine	Contains two programs to control the virtual leading axis: <u>pLineStartUp()</u> : Initialization of dynamic parametes of leading axis <u>pLineBackground()</u> : Control sequence to control leading axis
pMainSequence	Contains main sequence to control the automatic mode and the subsequences for the traversing winder and the line
pTravWinder	Consists of four programs to control the traversing winder: <u>pTravWinderStartUp()</u> : Initialization of the configuration parameter of the winder and the traverser. The set parameters of the traverser should not be overwritten each time the CPU reboots; as a consequence initialization is split up into a cyclic initialization and a one-off initialization. After the first initialization, the global retain bit "gaboTWTTravSetParamInitialized[]" is set to TRUE. As a consequence, only the cyclic data are subsequently initialized. If all of the parameters are to be reset, then the global bit must have been previously set to FALSE. <u>pTravWinderBackground()</u> Background program containing the sequence to control the traversing winder. There is a common control sequence to control the traverser and the winder. Additionally the main function block FBTravCtrl is called here and the IT-Diag data exchange is done here. <u>pTravWinderIPO()</u> Call of winder function block FBLConWinder in the IPO-cycle <u>pTravWinderMTCamCal()</u> Motion-Task program calling function block FBCalcPrio to calculate cams with traversing profile
pTravWinderHMI	Contains programs and functions to control the HMI functionality: <u>pTravWinderHMI()</u> : Control data exchange and functionality of the HMI <u>pTravWinderMTCheckCalc()</u> : Motion-Task program calling function blocks FCChkCalc and FBChkPrio to validate the traversing set parameters

Program sequence

The project example provided already includes control sequences for the individual technological modules - winder, traverser and leading axis. The following diagram clearly illustrates how the individual step sequences run:

Fig. 4-2 Program sequence



To control the corresponding blocks, each module (traversing winder, leading axis) has its own step sequence that is called in the background task. Within these step sequences, the corresponding input and output parameters of the blocks are accessed in order to guarantee a correct switch-on/switch-off sequence. The global switch-on and switch-off sequence is realized via the main sequence.

NOTE

The interconnection of the single function blocks can be seen in chapter [3.5](#)

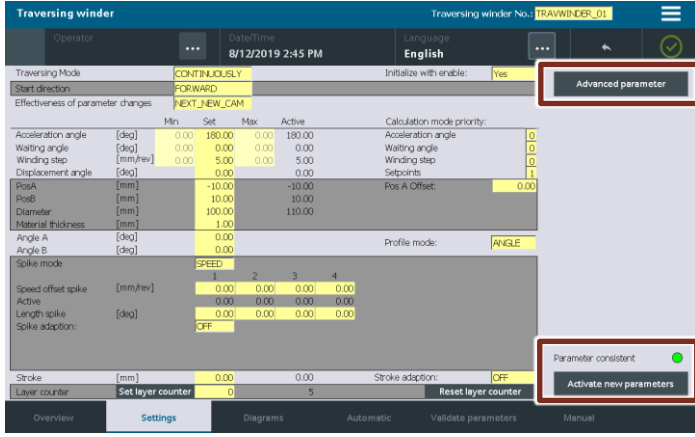
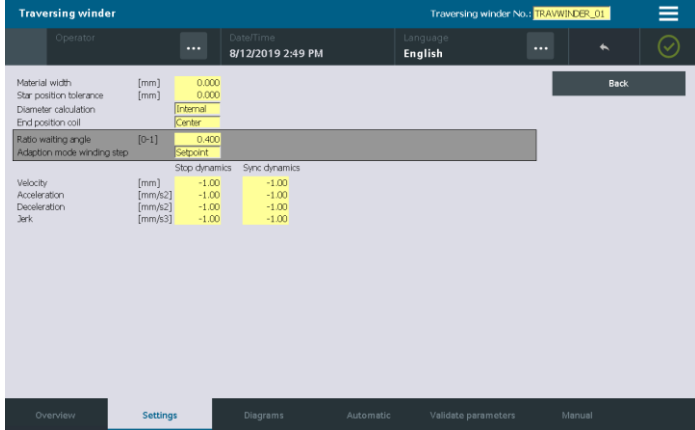
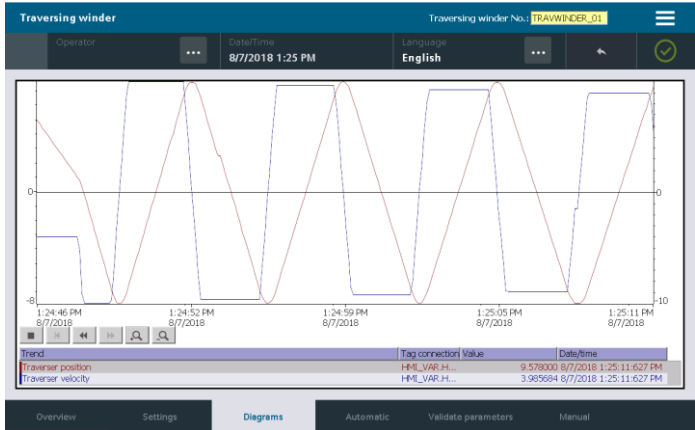
4.2 Description of the HMI

A HMI sample project is included in the project example, and can be used to control the traversing winder application. The HMI offers the following functions:

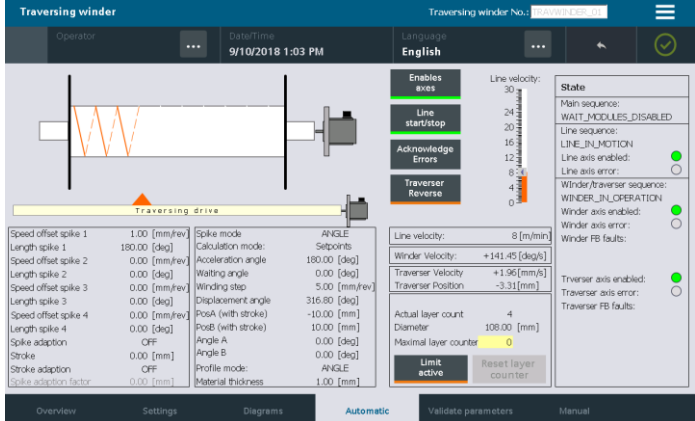
- Entering setpoints for the traverser application (sLTravSetParamType)
- Displaying actual values of the current profile calculation (sLTravActParamType)
- Manually controlling the traverser/winder axis
- Automatic mode
- Controlling and visualizing the parameter check (FCChkCalc, FBChkPrior)

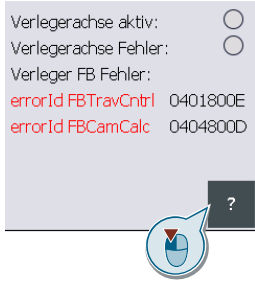
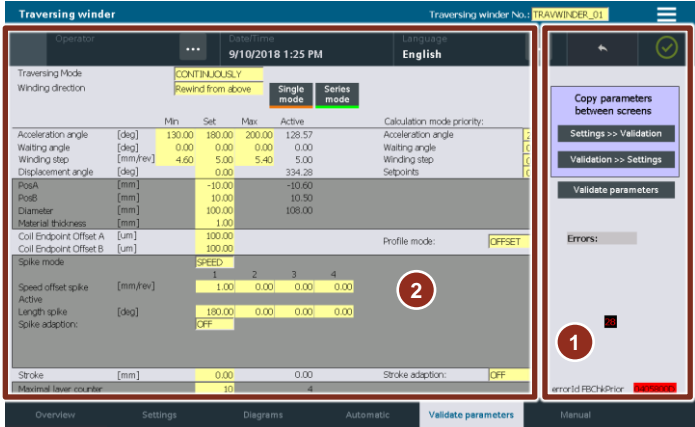
All of the variables required for the HMI are defined in the global unit dTravWinderHMI. The logic to control the HMI is implemented in the unit pTravWinderHMI. The single programs are called in the background tasks as well as in the Motion Task 3.

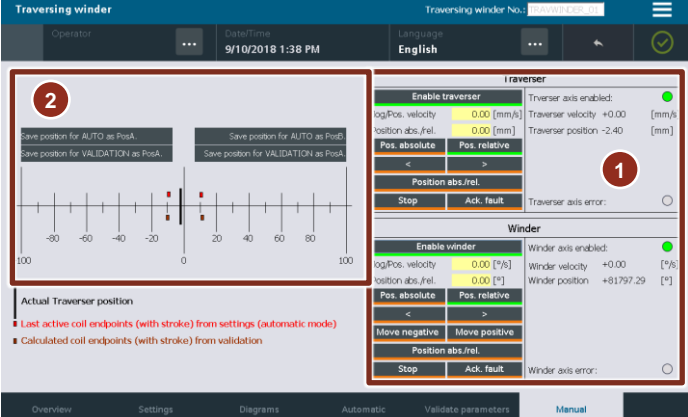
Table 4-3 Description of the HMI screens

Description	HMI screen
<p>Settings</p> <p>All of the traverser setpoints can be parameterized in this overview screen.</p>	 <p>Depending on the selected traversing mode, all setpoints can be entered here within the sTravSetParamType structure. When clicking on the "Activate new parameters" button, the modified parameters are copied to the setpoint structure of the selected traverser (traversing winder No.). If the parameters of the HMI and the parameters of the controller are identical this will be indicated by the "Parameter consistent" sign. Via the button "Advanced parameter" further settings can be done.</p>
<p>Advanced settings</p> <p>Further setpoints of the traverser can be parameterized in this screen. Via the button "Back" you can navigate back to the main settings</p>	
<p>Diagram</p> <p>The diagram provides a basic diagnostics function, and shows the actual traverser position and the traverser velocity.</p>	
<p>Automatic mode</p>	

4 Project example

Description	HMI screen
<p>The automatic mode of the traverser can be used in this screen.</p>	 <p>Traversing winder Traversing winder No.: 0000000000000000</p> <p>Operator: ... DateTime: 9/10/2018 1:03 PM Language: English</p> <p>Enables axes Line start/stop Acknowledge Errors Traverser Reverse</p> <p>Line velocity: 30 [m/min]</p> <p>State</p> <ul style="list-style-type: none"> Main sequence: WAIT_MODULES_DISABLED Line sequence: LINE_IN_MOTION Line axis enabled: <input checked="" type="checkbox"/> Line axis error: <input type="checkbox"/> Winder/traverser sequence: WINDER_IN_OPERATION Winder axis enabled: <input checked="" type="checkbox"/> Winder axis error: <input type="checkbox"/> Winder FB faults: <input type="checkbox"/> Traverser axis enabled: <input checked="" type="checkbox"/> Traverser axis error: <input type="checkbox"/> Traverser FB faults: <input type="checkbox"/> <p>Line velocity: 8 [m/min]</p> <p>Winder Velocity: +141.45 [deg/s]</p> <p>Traverser Velocity: +1.96 [mm/s]</p> <p>Traverser Position: -3.31 [mm]</p> <p>Actual layer count: 4</p> <p>Diameter: 108.00 [mm]</p> <p>Maximal layer counter: 0</p> <p>Limit active Reset layer counter</p> <p>Overview Settings Diagrams Automatic Validate parameters Manual</p>

Description	HMI screen
<p>For 1</p>	<p><u>Enable modules:</u> The main step sequence of the program is started, and the axis as well as the winder and traverser blocks is activated.</p> <p><u>Line start/stop:</u> The virtual leading axis is started and traversing begins. The web velocity can be entered in m/min using the slider on the right-hand side</p> <p><u>Acknowledge errors:</u> In the case of fault, active application faults can be acknowledged here.</p> <p><u>Traverser reverse:</u> Triggering the "reverseNow" input of the traverser. The traverser reverses within the active cam</p> <p><u>Limit active:</u> When activated, in the "Continuously" traversing mode, a maximum number of layers can be configured, at which the traverser will be stopped (Stop of line axis)</p> <p><u>Reset layer count:</u> The current number of layers is reset to zero</p>
<p>For 2</p>	<p>Visualization of the current status of the step sequences and the axes. If a fault occurs during processing, then the actual fault number is displayed along with the status of the subordinate blocks. A fault description can be displayed by clicking on the "?" button</p>  <p>Cam calculation error. Check the error output (errorID) of function block FB CamCalc.</p> <p>There is no solution for the specified parameters and modes.</p> <p>CLOSE</p>
<p>For 3</p>	<p>Currently calculated profile parameters of the traverser (s TravActParam)</p>
<p>Check parameters</p> <p>Plausibility check of the currently entered parameters.</p>	

Description	HMI screen
<p>For 1</p>	<p>The page structure is similar to that used to enter setpoint parameters. The parameter check function can be deployed using the buttons on the right-hand side:</p> <p><u>Setting >> Validation:</u> Copy the parameters from the setting page to the plausibility check</p> <p><u>Validation >> Setting:</u> If a correct setting was found, then the parameters can be copied back to the setting page as new setpoint</p> <p><u>Validate parameters:</u> Start the parameter validation. If a fault occurs when carrying out the check, then this is displayed using a number in the lower right-hand section (in this case, fault No. 28). A description of the fault is displayed by clicking on the number</p> <div data-bbox="671 689 1361 768" style="border: 1px solid black; padding: 5px;"> <p>28: ERROR The interval defined by the parameterized limit value does not allow a solution to be found in the actual calculation mode</p> <p style="text-align: center;">CLOSE</p> </div>
<p>For 2</p>	<p>Selecting the type of check and the setpoints to be checked</p> <p><u>Single mode:</u> Checks the parameters for a selected calculation mode and defined layer (FCChkCalc)</p> <p><u>Series mode:</u> Checks the parameters for a certain priority assignment of the calculation modes and a defined number of layers (FBChkPrior)</p>
<p>Manual mode</p> <p>The drives are manually traversed from the user interface of the application. Only available if the main step sequence is not active.</p>	
<p>For 1</p>	<p>Controlling the basic axis functions</p> <ul style="list-style-type: none"> • Enable axis • Jog • Positioning (relative/absolute) • Stop • Acknowledge fault
<p>For 2</p>	<p>Accept the actual traverser position as left-hand or right-hand coil edge point for checking the parameters or as traversing setpoint.</p>

4.3 Expansion to include additional traversing winder

The project example has been structured so that it can be very easily expanded to include further traversing winder stations. All of the required blocks – as well as the associated data structures – have already been created as array of length one up to constant NUM_OF_TRAVWINDER. Expanding the project example to include an additional traversing winder is described as example in the following.

Adapting the program

1. Create two additional technology objects for the winder and the traverser, and make the appropriate parameterization
2. Create two additional cams for the second traversing winder
3. Increase the user constant "NUM_OF_TRAVWINDER" to a value of "2" (pTravWinder -> Interface). By recompiling the blocks, all of the data arrays are expanded by the appropriate quantity.

Figure 4-3 Increase constant "NUM_OF_TRAVWINDERS"

```

30 //----- Device Global Constants -----
31 VAR_GLOBAL CONSTANT
32   NUM_OF_TRAVWINDERS      : USINT := 2;
33   //<<*** start label travwinder constant ***>>
34   TRAVWINDER_01 : USINT := 0;
35   TRAVWINDER_02 : USINT := 1;
36
37
38   // <<*** end label travwinder constant ***>>
39 END_VAR

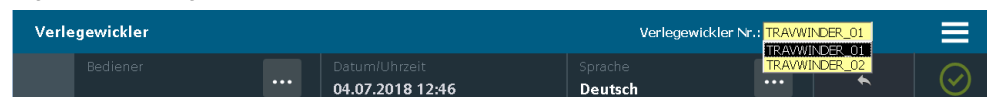
```

4. Copy the initialization for the already existing winder and traverser. Here, interconnect the new technology objects with the appropriate parameters (pTravWinderStartUp()).

Adapting the HMI

In order to switch the HMI between the two instances of the traversing winder, text list "traverserWinderSelection" must be increased with an additional entry and the required name and index "2" (text and graphic lists -> traverserWinderSelection). The corresponding instance can then be selected in the upper part of the HMI.

Fig. 4-4 Selecting the traverser instance



5 Functional description

5.1 Data types

Enumeration types

Type declarations of enumeration types are provided for some of the input and output parameters of the function blocks. Various modes and properties can be pre-set using these parameters.

Data arrays

Type declarations as data arrays are made available for certain input or output parameters of the function parameters.

Data structures

The function blocks of the traverser application are partially parameterized using data structures. These should be set-up for the particular block. A structure with the corresponding parameters is provided for each block.

5.2 Enumeration types

Table 5-1 Enumeration types

Enumeration types	Contents
eLTravDirectionType	Defines the actual or the start direction of the traversing axis Declared in: LTravLib.dType
eLTravCalcModeType	Specifies the calculation mode for the plausibility check of the parameters and the cam calculation Declared in: LTravLib.dType
eLTravCalcPModeType	Defines the calculation modes Declared in: LTravLib.dType
eLTravCoilModeType	Defines the coil profile modes Declared in: LTravLib. dType
eLTravSpikeModeType	Defines the spike modes Declared in: LTravLib. dType
eLTravWindingModeAType	Defines the winding direction Declared in: LTravLib. dType
eLTravParameterChangeModeType	Defines when parameter changes become effective Declared in: LTravLib. dType
eLTravTraversingModeType	Defines the traversing mode Declared in: LTravLib.dType
eLTravDefEndPosCoilType	Defines the end points A and B of the coil Declared in: LTravLib.dType
eLTravCalcReasonType	Defines which cam is calculated Declared in: LTravLib.dType
eLTravProfileSegmentType	Defines the actual segment of the cam Declared in: LTravLib.dType
eLTravEnableBehaviorType	Defines the behavior at rising edge on input enable. Declared in: LTravLib.dType
eLTravWindingStepAdaptionType	Defines the winding step adaption mode. Declared in: LTravLib.dType

eLTravDirectionType

This data type is used to define the start direction of the traversing axis.

Table 5-2 eLTravDirectionType

Element	Description
FORWARD	Start direction, actual direction, forwards
BACKWARD	Start direction, actual direction, backwards
LASTDIRECTION	Start in the last direction that was active

eLTravCalcModeType

This data type specifies the calculation mode of the cam or the parameter plausibility check.

Table 5-3 eLTravCalcModeType

Element	Description
ACCELERATION_ANGLE	Cam calculation or plausibility check with adaptation of the acceleration angle
WAITING_ANGLE	Cam calculation or plausibility check with adaptation of the waiting angle
WINDING_STEP	Cam calculation or plausibility check with adaptation of the winding step
SETPOINTS	Cam calculation or plausibility check after setpoint input

eLTravCalcPModeType

Internal data type for cam calculations or the plausibility check if several calculation modes are assigned priorities.

Table 5-4 eLTravCalcPModeType

Element	Description
ACCELERATION_ANGLE	Cam calculation or plausibility check with adaptation of the acceleration angle
WAITING_ANGLE	Cam calculation or plausibility check with adaptation of the waiting angle
WINDING_STEP	Cam calculation or plausibility check with adaptation of the winding step
SETPOINTS	Cam calculation or plausibility check after setpoint input
NA	No valid mode

eLTravCoilModeType

This data type specifies the definition mode of the coil profile.

Table 5-5 eLTravCoilModeType

Element	Description
DEFINE_WITH_ANGLE	The coil profile is defined using the coil edge angle.
DEFINE_WITH_OFFSET	The coil profile is defined using a layer offset.

eLTravSpikeModeType

This data type specifies the definition mode of spikes.

Table 5-6 eLTravSpikeModeType

Element	Description
DEFINE_AS_SPEED_OFFSET	The spike is defined by the spike length and the additional velocity
DEFINE_AS_POS_OFFSET	The spike is defined by the spike length and the position offset of the traverser

eLTravWindingModeAType

The winder type is parameterized using this enum.

Table 5-7 eLTravWindingModeAType

Element	Description
UNWIND_FROM_ABOVE	Unwinding from the top
UNWIND_FROM_BELOW	Unwinding from the bottom
REWIND_FROM_ABOVE	Winding from the top
REWIND_FROM_BELOW	Winding from the bottom

eLTravParameterChangeModeType

The effectiveness of parameter changes is set using this enum.

Table 5-8 eLTravParameterChangeModeType

Element	Description
NEXT_NEW_CAM	Changed parameters are used in the next cam calculation and activated with its synchronization (AT_THE_END_OF_CAM_CYCLE). If the cam to be synchronized is already calculated the activation is delayed by one cycle. Behaviour as before V302.
NEXT_CAM	Changed parameters are used in the next cam calculation. The pending cam will be recalculated.
IMMEDIATELY	Based on the changed parameters a new cam is calculated and switched immediately.

eLTravTraversingModeType

Mode of traversing is parameterized using this enum.

Table 5-9 eLTravTraversingModeType

Element	Description
CONTINUOUSLY	Continuously traversing with spikes and stroke
STEP_WIND	Stepwise traversing profile

eLTravDefEndPosCoilType

The endpoints are defined using this enum.

Table 5-10 eLTravDefEndPosCoilType

Element	Description
CENTER	Endpoints are defined as center of outside layer
OUTSIDE	Endpoint A is defined as left outside of coil endpoint B is defined as right outside of coil

eLTravCalcReasonType

This enum is used as handshake between the function blocks FBTravCntrl and FBCamCalc. It is used to define which cam should be calculated.

Table 5-11 eLTravCalcReasonType

Element	Description
NEXT_CAM	Just the next cam will be calculated
RECALC_ACTUAL_CAM	Actual cam will be new calculated
RECALC_ACTUAL_CAM_AND_NEXT_CAM	Actual and next cam will be new calculated
RECALC_NEXT_CAM	Next cam will be new calculated
INITIALIZATION	Initialization for function block CamCalc at rising edge on enable input at function block FBTravCntrl

eLTravProfileSegmentType

This enum shows the actual segment of the traverser in the cam profile. The cam is separated in 4 segments, two in forward and two in backward direction.

Table 5-12 eLTravProfileSegmentType

Element	Description
Segment_1	forward segment of last traversing cycle
Segment_2	backward segment of last traversing cycle
Segment_3	forward segment of actual traversing cycle
Segment_4	backward segment of actual traversing cycle

eLTravEnableBehavior

This enum defines the behavior while rising edge on input enable of the function block FBTravCntrl. Depending on this setting the actual parameter are used to generate the new profile or all values will be initialized.

Table 5-13 eLTravEnableBehavior

Element	Description
NO_INITIALIZATION	Actual parameter are used. A profile compatible to the actual parameters will be generated.
INITIALIZE_PROFILE	Actual parameters will be initialized. Profile will be generated based on the set values. The layer counter will be reset.

eLTravWindingStepAdaptionModeType

This enum defines the mode in which the winding step will be adapted – in adaption mode winding step to hold a given displacement angle.

Table 5-14 eLTravWindingStepAdaptionModeType

Element	Description
MINIMUM_BASED	The winding step will be detected based on minimum value upwards.
SETPOINT_BASED	The winding step will be detected based on the setpoint up- and downwards. The nearest value will be set.

5.3 Array data types

Table 5-15 Array data types

Data type	Description
aLTravErrorArrayType	This array includes the fault and warning (alarm) messages of the plausibility check. Declared in: LTravLib.dType
aLTravModePriorityType	The priority assignment of the calculation modes are parameterized in this array. Declared in: LtravLib.dType

aLTravErrorArrayType

This array contains the fault and warning (alarm) messages of the plausibility check.

Table 5-16 aLTravErrorArrayType

I/O	Element	Data type	Description
OUT	aLTravErrorArrayType	ARRAY [0..LTRAVLIB_NUM_OF_ERROR-1] OF BOOL	Every element in this array refers to an error or warning (alarm) message of the plausibility check.

aLTravModePriorityType

Each element in this array defines the priority of a calculation mode for the cam calculation or the plausibility check. The value of an element defines the priority. 0 means that the corresponding calculation mode is not used. The priority for the individual calculation modes must have different values.

Table 5-17 aLTravErrorArrayType

I/O	Element	Data type	Description
IN	aLTravi8ModePriorityType	ARRAY [0..LTRAVLIB_NUM_OF_CALCMODE-1] OF SINT -1] OF BOOL	Priority assignment of the calculation modes: ARRAY[0]: Acceleration angle ARRAY[1]: Waiting angle ARRAY[2]: Winding step ARRAY[3]: Setpoints

5.4 Data structures

Table 5-18 Data structures

Data type	Description
sTravSetParamType	This data structure contains all of the data relevant for traverser operation and calculating the motion profile of the traverser. Declared in: LTravLib.dType
sTravActParamType	This structure contains all layer-dependent parameters and is also used for transferring data between blocks. Declared in: LTravLib.dType
sTravActChkParamType	Return data structure of the FCChkCalc function. The structure contains all of the parameters and diagnostic variables relevant for the motion profile. Declared in: LTravLib.dType
sTravParamType	This structure encompasses the sTravSetParamType and sTravActParamType structures. Declared in: LTravLib.dType
sDynamicsType	This structure contains the dynamic parameters used for synchronization and de-synchronization and for stopping the traver.

The following generally applies for designating/naming the structure elements

- [IN] Values that the user must provide
- [OUT] Results or feedback signals
- [IO] Values, which depending on how the block is interconnected, are supplied by the user or written by the function.

sDynamicsType

These parameters can be set by the user. All parameters have default values. If these values will not be changed the values are compatible to older versions.

Table 5-19 sLTravDynamicsType

I/O	Element	Data type	Description
IN	r64Velocity	LREAL	<p>Transition velocity for synchronization and de-synchronization of the cam.</p> <p>Value > 0: given value is used Value = 0: not valid Value < 0: the value set in the system variable userDefault.synchDynamics.velocity of the connected following object is used.</p> <p>Default value: -1.0</p>
IN	r64Acceleration	LREAL	<p>Command acceleration for synchronization and de-synchronization of the cam.</p> <p>Value > 0: given value is used Value = 0: not valid Value < 0: the value set in the system variable userDefault.synchDynamics.positiveAccel of the connected following object is used.</p> <p>Default value: -1.0</p>
IN	r64Deceleration	LREAL	<p>Command deceleration for synchronization and de-synchronization of the cam.</p> <p>Value > 0: given value is used Value = 0: not valid Value < 0: the value set in the system variable userDefault.synchDynamics.negativeAccel of the connected following object is used.</p> <p>Default value: -1.0</p>
IN	r64Jerk	LREAL	<p>Jerk for the acceleration and deceleration.</p> <p>Value > 0: given value is used Value = 0: trapezoidal profile is used Value < 0: the valuse set in the system variables: userDefault.syncDynamics.positiveAccelStartJerk userDefault.syncDynamics.positiveAccelEndJerk userDefault.syncDynamics.negativeAccelStartJerk and userDefault.syncDynamics.negativeAccelEndJerk of the connected following object are used.</p> <p>Default value: -1.0</p>

sTravSetParamType

The user sets all parameters of this structure.

Table 5-20 sTravSetParamType

I/O	Element	Data type	Description
IN	eCoilMode	eCoilModeType	DEFINE_WITH_ANGLE: Definition of the coil edge profile using the angle DEFINE_WITH_OFFSET: Definition of the coil edge profile using the layer offset
IN	r32CoilAngA	REAL	This parameter defines the coil profile in position A (r64StartPosA): Coil profile, mode Coil edge angle: eCoilMode = DEFINE_WITH_ANGLE: [°] The coil profile is defined using the coil edge angle. The coil becomes wider for positive angles and narrower for negative angles. The absolute value of the angle must be less than 90°. Coil profile mode Layer offset: eCoilMode = DEFINE_WITH_OFFSET: [um] The coil profile is defined using a layer offset. The coil becomes wider for positive values – and narrower for negative values.
IN	r32CoilAngB	REAL	This parameter defines the coil profile in position B (r64StartPosB): Coil profile mode Coil edge angle: eCoilMode = DEFINE_WITH_ANGLE: [°] The coil profile is defined using the coil edge angle. The coil becomes wider for positive angles – and narrower for negative angles. The absolute value of the angle must be less than 90°. Coil profile mode Layer offset : eCoilMode = DEFINE_WITH_OFFSET: [um] The coil profile is defined using a layer offset. The coil becomes wider for positive angles – and narrower for negative angles.
IN	r32MatThick	REAL	[mm] Material thickness. This value must be positive. Default value: 1.0 mm.
IN	r32MatWidth	REAL	[mm] Material width for the calculation of the effective material thickness. If not used, effective thickness is the value specified in r32MatThick. Default value: 0.0 mm.
IN	r32StartPosA	REAL	[mm] This parameter defines the initial coordinate of the lefthand coil edge. The stroke is not to be taken into account here. The value must be less than the coordinate of the righthand edge point. The distance between the coil edges must be greater than the winding step. Default value: -10.0 mm.

5 Functional description

I/O	Element	Data type	Description
IN	r32StartPosB	REAL	[mm] This parameter defines the initial coordinate of the righthand coil edge. The value must be greater than r64StartPosA. The distance between the coil edges must be greater than the winding step. Default: 10.0 mm.
IN	r32SetDisplAng	REAL	[°] Displacement angle All positive values are permitted; for values over 360°, the corresponding modulo value is determined. Abbreviation: DA
IN	eSpikeMode	eSpikeMode Type	DEFINE_AS_SPEED: Spike mode Additional velocity, the spike is defined using the velocity offset and the spike length. DEFINE_AS_POS_OFFSET: Spike mode position offset, the spike is defined using the position offset and length.
IN	r32AddS1	REAL	Spike mode, additional velocity: [mm/rev] Spike over-velocity of the first spike in the traversing cycle. Spike mode, position offset: [mm] Position offset that is generated by the first spike. The spike function is deactivated with 0.0. Abbreviation: AS1
IN	r32AddLength1	REAL	[°] Length of the first spike. The spike length is specified, referred to the master axis (winder axis or machine encoder). All positive values are permitted; the spike is deactivated if 0 is parameterized. Abbreviation: AS1C
IN	r32AddS2	REAL	Spike mode additional velocity: [mm/rev] Spike over-velocity of the second spike in the traversing cycle. Spike mode, position offset: [mm] Position offset that is generated by the second spike. The spike function is deactivated with 0.0. Abbreviation: AS2
IN	r32AddLength2	REAL	[°] Length of the second spike. The spike length is specified, referred to the master axis (winder axis or machine encoder). All positive values are permitted; the spike is deactivated if 0 is parameterized. Abbreviation: AS2C

5 Functional description

I/O	Element	Data type	Description
IN	r32AddS3	REAL	<p>Spike mode, additional velocity: [mm/rev] Spike over-velocity of the fourth spike in the traversing cycle.</p> <p>Spike mode, position offset: [mm] Position offset that is created by the third spike.</p> <p>The spike function is deactivated with 0.0.</p> <p>Abbreviation: AS3</p>
IN	r32AddLength3	REAL	<p>[°] Length of the third spike.</p> <p>The length of the spike is specified referred to the master axis (winder axis or machine encoder).</p> <p>All positive values are permitted. The spike is deactivated if 0 is parameterized.</p> <p>Abbreviation: AS3C</p>
IN	r32AddS4	REAL	<p>Spike mode, additional velocity: [mm/rev] Spike over-velocity of the first spike in the traversing cycle.</p> <p>Spike mode, position offset: [mm] Position offset that is created by the fourth spike.</p> <p>The spike function is deactivated with 0.0.</p> <p>Abbreviation: AS4</p>
IN	r32AddLength4	REAL	<p>[°] Length of the fourth spike.</p> <p>The length of the spike is specified referred to the master axis (winder axis or machine encoder).</p> <p>All positive values are permitted. The spike is deactivated if 0 is parameterized.</p> <p>Abbreviation: AS4C</p>
IN	boAdaptSpike	BOOL	<p>FALSE: Spike adaptation deactivated</p> <p>TRUE: Spike adaptation activated</p>
IN	r32SpikeACoeff	REAL	<p>[mm] Correction value of the spike adaptation in the spike mode, position offset.</p> <p>All positive values are permitted; adaptation is disabled with 0.</p>
IN	eCalcMode	eCalcMode Type	<p>[-] Calculation mode:</p> <p>ACCELERATION_ANGLE: Cam calculation and/or plausibility check with adaptation of the acceleration angle</p> <p>WAITING_ANGLE: Cam calculation and/or plausibility check with adaptation of the waiting angle</p> <p>WINDING_STEP: Cam calculation and/or plausibility check with adaptation of the winding step</p> <p>SETPOINTS: Cam calculation and/or plausibility check after setpoint input</p>

5 Functional description

I/O	Element	Data type	Description
IN	r32SetAccAng	REAL	[°] Setpoint acceleration angle The acceleration angle is configured, referred to the master axis (winder axis, machine encoder). In this case, the spikes are not taken into account. All positive values can be entered. The parameter is only used if eCalcMode is not equal to ACCELERATION_ANGLE. Default: 180.0. Abbreviation: AA
IN	r32MinAccAng	REAL	[°] Minimum acceleration angle The parameter sets the lower limit to calculate the acceleration angle in the mode eCalcMode = ACCELERATION_ANGLE. All positive values can be entered.
IN	r32MaxAccAng	REAL	[°] Maximum acceleration angle This parameter sets the upper limit to calculate the acceleration angle in the mode eCalcMode = ACCELERATION_ANGLE. All positive values can be entered.
IN	r32SetWaitAng	REAL	[°] Setpoint, waiting angle This parameter defines the setpoint for the waiting angle. The waiting angle is configured referred to the master axis (winder axis, machine encoder). All positive values can be entered. The parameter is only used if eCalcMode is not equal to WAITING_ANGLE. Abbreviation: WA
IN	r32MinWaitAng	REAL	[°] Minimum waiting angle This parameter sets the lower limit to calculate the waiting angle in the mode eCalcMode = WAITING_ANGLE. All positive values can be entered.
IN	r32MaxWaitAng	REAL	[°] Maximum waiting angle This parameter sets the upper limit to calculate the waiting angle in the mode eCalcMode = WAITING_ANGLE. All positive values can be entered.
IN	r32RatWaitAng	REAL	[0..1] Ratio between waiting angle at position A and the sum waiting angle. Default: 0.5.
IN	r32SetWindStep	REAL	[mm/rev] Winding step This parameter defines the WINDING_STEP of the traverser, i.e. the feed of the traversing axis referred to one winder revolution if the calculation mode is not equal to winding step. The parameter includes the offset of the winding layers as well as the material width. All positive values can be entered. Abbreviation: WS Default value: 5.0

5 Functional description

I/O	Element	Data type	Description
IN	r32MinWindStep	REAL	[mm/rev] Minimum winding step This parameter sets the lower limit to calculate the winding step in the mode eCalcMode = WINDING_STEP. All positive values can be entered.
IN	r32MaxWindStep	REAL	[mm/rev] Maximum winding step This parameter sets the upper limit to calculate the winding step in the mode eCalcMode = WINDING_STEP. All positive values can be entered.
IN	r32Stroke	REAL	[mm] Stroke This parameter defines the initial stroke beyond the coil end points. All positive values can be entered. The function is not active if 0.0 is parameterized.
IN	boAdaptStroke	BOOL	FALSE: Layer dependent adaptation of the stroke deactivated TRUE: Layer dependent adaptation of the stroke activated
IN	r32CoreDiameter	REAL	[mm] Coil core diameter All positive values can be entered. Default: 100.0.
IN	boExternDiameter	BOOL	FALSE: The internal diameter calculation is active, based on the coil core diameter, the layer counter and the material thickness. TRUE: The external diameter value is used (sTravActParam) Default value: False
IN	r32TolStartPos	REAL	[mm] Tolerance bandwidth to adapt the end positions of the cam for encoder-dependent position actual value fluctuations. Only positive values are permitted.
IN	sStopDynamics	sDynamics Type	Contains the Dynamic values used to stop the axis.
IN	sSyncDynamics	sDynamics Type	Contains the Dynamic values used for synchronization and de-synchronization
IN	eDefEndPosCoil	eDefEndPos CoilType	Defines the endpoints of the coil. CENTER: Position A is in the middle of the most left layer OUTSIDE: Position A is at the left outside of the most left layer, position B is at the right side of the most right layer. Default value: CENTER
IN	eTraversingMode	eTraversing ModeType	Defines the mode of the traverser CONTINUOUSLY: continuously traversing STEP_WIND: stepwise traversing Default value: CONTINUOUSLY

5 Functional description

I/O	Element	Data type	Description
IN	u16NumberOfSteps	UINT	Defines the number of steps for step wise traversing. This parameter is just used in mode STEP_WIND. Default value: 0
IN	r32LayerPerStep	REAL	Defines the number of layers to be wound in one step. The resulting angle will be additionally calculated with the parameter r32SetWaitAng Default value: 0.0
IN	r64StartPosOffset	LREAL	Defines a fixed offset for Position A and B Default value: 0.0
IN	eLTravEnable Behavior	eLTravEnable BehaviorType	Defines the behavior at rising edge on enable. NO_INITIALIZATION: It will be synchronized to saved profile at falling edge. INITIALIZE_PROFILE: A new profile with set parameters will be generated. Actual values will be reset also. Default value: INITIALIZE_PROFILE
IN	r32StepWindWait Ang	REAL	Waiting angle for mode step wind. This angle is used just in positions A and B analogue to the waiting angle in mode continuously.
IN	eWindingStep AdaptionMode	eLTravWindingStepAdaption ModeType	Defines the base and the mode the winding step adaption value will be calculated by.

sTravActParamType

Elements of the data structure sTravActParamType:

Table 5-21 sTravActParamType

I/O	Element	Data type	Description
IO	toSelectedCam	CAMTYPE	TO reference of the cam, which is to be used to calculate the motion profile
IO	eWindingMode	eLTravWindingModeAType	[-] Current active winding mode (will be updated with a rising edge of the enable input of the FBTravCntl)
IO	i32LayerCount	DINT	[-] Actual value of the layer counter The user can pre-assign this value – e.g. for the case that the traverser is to be used for unwinding
OUT	eActDirection	eDirection Type	Actual direction of motion of the traverser
IO	i16ActLayerOffset	INT	[-] Variable for the layer-dependent profile

5 Functional description

I/O	Element	Data type	Description
OUT	r64ActDiameter	LREAL	[mm] If an external diameter value is not used (sTravSetParam.boExternDiameter = FALSE) then the actually calculated diameter is output in this variable. If an external diameter value is used (sTravSetParam.boExternDiameter = TRUE), the value of the external diameter calculation must be linked with this variable. The actual diameter will be rounded internally using the core diameter and the effective material thickness.
IO	r64ActPosA	LREAL	[mm] Actual coordinate of the lefthand coil edge – including the layer-dependent factors.
IO	r64ActPosB	LREAL	[mm] Actual coordinate of the righthand coil edge – including the layer dependent factors.
OUT	r32ActWindStep	REAL	[mm/rev] Actual winding step including the layer dependent factors and adaptations.
OUT	r32ActAccAng	REAL	[°] Actual acceleration angle including the layer dependent factors and adaptations.
OUT	r32ActWaitAng	REAL	[°] Actual waiting angle including the layer dependent factors and adaptations.
OUT	r32ActDisplAng	REAL	[°] This variable indicates the calculated offset angle. If the calculation mode is not SETPOINTS, then the value is identical with the setpoint of the displacement angle calculation.
OUT	r32ActStroke	REAL	[mm] Actual stroke
OUT	r32ActAddS1	REAL	[mm/rev mm] Actual value of the spike velocity of the first spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the first spike for the spike mode, position offset.
OUT	r32ActAddS2	REAL	[mm/rev mm] Actual value of the spike velocity of the second spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the second spike for the spike mode, position offset.
OUT	r32ActAddS3	REAL	[mm/rev mm] Actual value of the spike velocity of the third spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the third spike for the spike mode, position offset.
OUT	r32ActAddS4	REAL	[mm/U mm] Actual value of the spike velocity of the fourth spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the fourth spike for the spike mode, position offset.

5 Functional description

I/O	Element	Data type	Description
OUT	r32EffThick	REAL	[mm] Effective material thickness. With r32MatWidth = 0.0 equals to r32MatThick. Otherwise it is calculated using the winding step, material thickness and material width.
IO	r32ActRevPoint	REAL	[°] Reversal point within the cam of the actual cycle.
IO	eCalcReason	eCalcReason Type	Handshake between FBTravCntrl and FBCamCalc
OUT	r32LayerGap	REAL	Gap between the discs in mode STEP_WIND
OUT	u16ActualStep	UINT	Actual step in mode STEP_WIND
OUT	r64LayersActual Step	LREAL	Actual number of layers wound in actual step
IO	r64DesyncPosition	LREAL	Master position in Cam at desynchronization. (disable FB, change parameters)
IO	r64SlaveDesync Position	LREAL	Slave position in Cam at desynchronization (disable FB, change parameters)
OUT	r64NextPosA	LREAL	[mm] Coordinate of the left hand coil edge of the next active cam
OUT	r64NextPosB	LREAL	[mm] Coordinate of the right hand coil edge of the next active cam
OUT	boChange Immediately	BOOL	Handshake to calculate steady profile
OUT	r32LastRevPoint	REAL	[°] Reversal point within the cam of the last cycle
OUT	r32CenterPoint	REAL	[°] Center point within the current calculated cam
OUT	eProfileSegment	eProfile SegmentType	Handshake of actual profile segment to calculate steady profile and detect synchronization point
OUT	r64StepWindPosB MasterPosition	LREAL	Synchronization point for position B in mode step wind. This value will be used just at first start and at start with eLTravEnableBehavior = INITIALIZE_PROFILE and the traverser position equals pos B.

sTravParamType

Elements of the data structure sTravParamType:

Table 5-22 sTravParamType

I/O	Element	Data type	Description
IN	sSet	sTravSetParamType	Refer to the definition for sTravSetParamType
I/O	sAct	sTravActParamType	Refer to the definition for sTravActParamType

sTravActChkParamType

Elements of the data structure sTravActChkParamType:

Table 5-23 sTravActChkParamType

I/O	Element	Data type	Description
OUT	i32ActLayerCount	DINT	[-] Actual value of the layer counter for the plausibility check
OUT	eActWindingMode	eWindingModeAType	Actual selection of the calculation mode for the plausibility check
OUT	r32ActDiameter	REAL	[mm] Actual coil diameter For the internal diameter calculation, the actually calculated diameter is displayed here – for an external diameter calculation, the external diameter value is rounded using the core diameter and the material thickness.
OUT	r32ActPosA	REAL	[mm] Actual coordinates of the lefthand coil edge (position A) including the optional adaptations.
OUT	r32ActPosB	REAL	[mm] Actual coordinates of the righthand coil edge (position B) including the optional adaptations.
OUT	r32ActWindStep	REAL	[mm/rev] Actual winding step If the calculation mode is not WINDING_STEP, then the setpoint of the winding step is displayed here.
OUT	r32ActAccAng	REAL	[°] Actual acceleration angle If the calculation mode is not ACCELERATION_ANGLE, the setpoint of the acceleration angle is displayed here.
OUT	r32ActWaitAng	REAL	[°] Actual waiting angle If the calculation mode is not WAITING_ANGLE, the setpoint of the waiting angle is displayed here.
OUT	r32ActDisplAng	REAL	[°] Actual displacement angle If the calculation mode is SETPOINTS, the resulting displacement angle is displayed; for other calculation modes, the value corresponds to the setpoint displacement angle.
OUT	r32ActStroke	REAL	[mm] Actual stroke The value includes the adaptation using the optional adaptations.

5 Functional description

I/O	Element	Data type	Description
OUT	r32ActAddS1	REAL	[mm/rev mm] Actual value of the spike velocity of the first spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the first spike for the spike mode, position offset
OUT	r32ActAddS2	REAL	[mm/rev mm] Actual value of the spike velocity of the second spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the second spike for the spike mode, position offset.
OUT	r32ActAddS3	REAL	[mm/rev mm] Actual value of the spike velocity of the third spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the third spike for the spike mode, position offset
OUT	r32ActAddS4	REAL	[mm/rev mm] Actual value of the spike velocity of the fourth spike of the traversing cycle for the spike mode, additional velocity. [mm] Actual position offset of the fourth spike for the spike mode, position offset
OUT	r32EffThick	REAL	[mm] Effective material thickness. With r32MatWidth = 0.0 equals to r32MatThick. Otherwise it is calculated using the winding step, material thickness and material width.
OUT	aboErrorArray	aErrorArrayType	[-] Output array of the plausibility check with error and warning information
OUT	boError	BOOL	[-] TRUE: User parameter error. FALSE: No user parameter error
OUT	boWarning	BOOL	[-] TRUE: Warning information for the user parameter FALSE: No warning information

5.5 FBTravCtrl

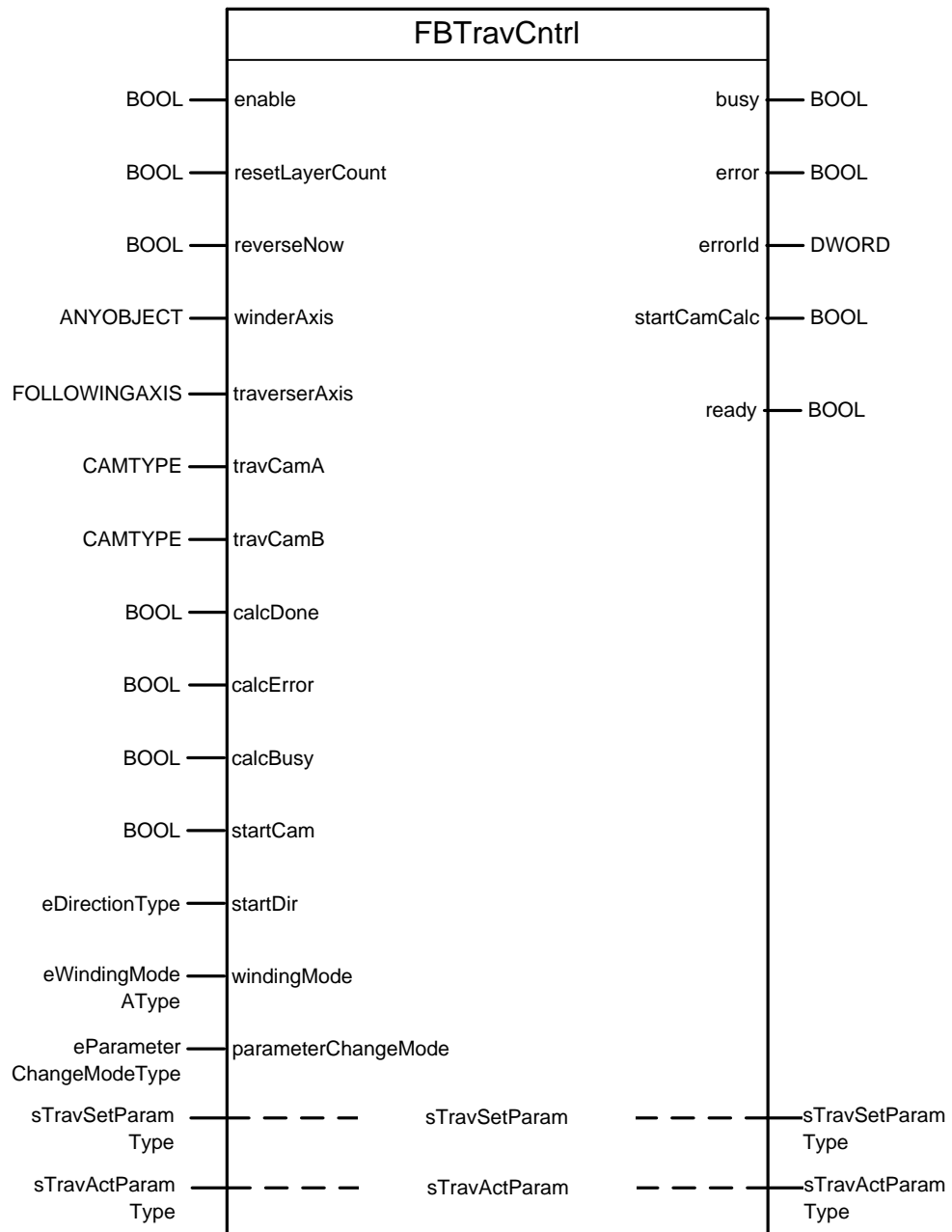
Task

Function block to control the traversing functionality.

The function block must be called in a cyclic task (Background, Servo, IPO, IPO_2 or Timer).

Schematic LAD Representation

Figure 5-1 FBTravCtrl



5.5.1 Input and output parameters

Input parameters

Table 5-24 Input parameters

Name	Data type	Initial value	Description
enable	BOOL	FALSE	TRUE: The FB is activated FALSE: The FB is deactivated Rising edge: Start of the initialization sequence Falling edge: Start of the stopping sequence
resetLayerCount	BOOL	FALSE	Rising edge: The layer counter is reset. The traversing profile is reinitialized; layer dependent functions (adaption, coil profile) are reset to initial values.
reverseNow	BOOL	FALSE	Rising edge: The traversing axis inverts its current direction of motion. If a cam change is not required, or the traversing arm reverses when moving forwards, then the application jumps within the active cam. For reverse motion, the application jumps into the new cam.
winderAxis	ANYOBJECT	TO#NIL	TO reference for the master axis (winder axis as positioning axis or machine encoder) The activation of the object must be programmed in the user program.
traverserAxis	FOLLOWING AXIS	TO#NIL	TO reference of the traversing axis The traversing axis must be a synchronous axis that is configured for cam synchronous operation (camming) with the winder axis using the cam objects configured in travCamA and travCamB. The activation of the object must be programmed in the user program.
travCamA	CAMTYPE	TO#NIL	TO reference for the first cam object
travCamB	CAMTYPE	TO#NIL	TO reference for the second cam object
calcDone	BOOL	FALSE	TRUE: Feedback signals that cam generation was completed error-free (FBCamCalc).
calcError	BOOL	FALSE	TRUE: Indicator that an error occurred during cam generation (FBCamCalc).
calcBusy	BOOL	FALSE	TRUE: Cam calculation in FBCamCalc is active

5 Functional description

Name	Data type	Initial value	Description
startCam	BOOL	FALSE	TRUE: The cam reference in travCamB is used as the first cam if the function is started. FALSE: The cam reference in travCamA is used as first cam if the function is started.
startDir	eDirectionType	FORWARD	FORWARD: The direction of motion of the traverser axis when starting the function is defined to be forwards. BACKWARD: The direction of motion of the traverser axis when starting the function is defined to be backwards. LASTDIRECTION: When starting the function, the last active direction of motion is accepted, otherwise the axis is started in the forwards direction.
windingMode	eWindingMode AType	REWIND_ FROM_ ABOVE	<u>Winder type</u> UNWIND_FROM_ABOVE: unwinding from the top UNWIND_FROM_BELOW: unwinding from the bottom REWIND_FROM_ABOVE: winding from the top REWIND_FROM_BELOW: winding from the bottom
parameterChange Mode	eParameter ChangeMode Type	NEXT_ NEW_CAM	NEXT_NEW_CAM: Changed parameters are used in the next cam calculation and activated with its synchronization (AT_THE_END_OF_CAM_CYCLE). If the cam to be synchronized is already calculated the activation is delayed by one cycle. Behaviour as before V302. NEXT_CAM: Changed parameters are used in the next cam calculation. The pending cam will be recalculated. IMMEDIATELY: Based on the changed parameters a new cam is calculated and switched immediately.

Input/output parameters

Table 5-25 Input/output parameters

Name	Data type	Initial value	Description
sTravActParam	sTravActParam Type		Details on this structure are included in the data type description
sTravSetParam	sTravSetParam Type		Details on this structure are included in the data type description

Output parameters

Table 5-26 Output parameters

Name	Data type	Initial value	Description
startCamCalc	BOOL	FALSE	TRUE: The FBCamCalc call to calculate the cam must be active. Rising edge: Block or motion task to calculate the cam must be started Falling edge: Cam calculation is interrupted
busy	BOOL	FALSE	TRUE: FB is active FALSE: FB is deactivated
error	BOOL	FALSE	TRUE: An error has occurred FALSE: No error Not activated for warnings.
errorId	DWORD	16#0000_0000	Error identification For warning messages, an error identification is output, however, the error output is not set.
ready	BOOL	FALSE	TRUE: Traverser is synchronized to the master FALSE: Traverser is not synchronized

5.5.2 Functionality

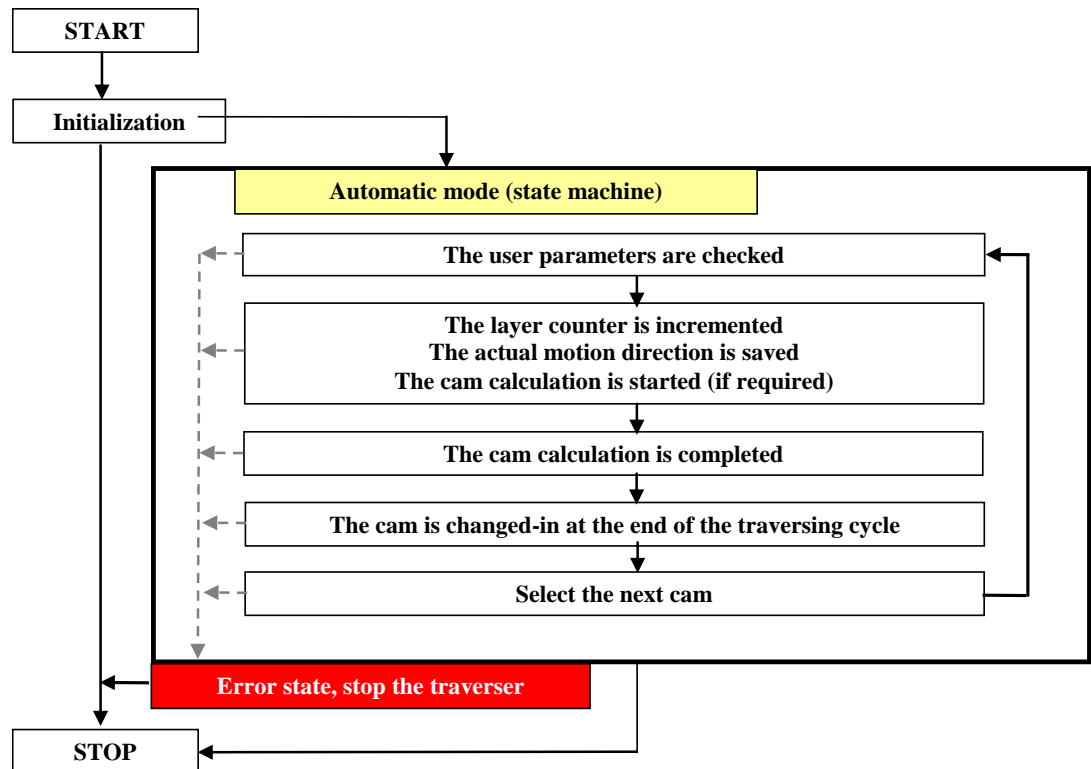
The function block includes the main program of the traversing application. The function is started by a rising edge at the *enable* input. After activation, the user data is checked and the state machine started. The cam calculation, cam change, direction change, layer counter and diameter calculation are performed in the state machine. The state machine is brought into the error status if errors occur during processing.

The direction in which the traverser (traversing arm) starts, the use of the cam as start cam and the layer counter can be controlled using the specified inputs. The motion profile of the traverser is calculated in a separate block (*FBCamCalc*) depending on the parameters. The corresponding control inputs are provided in order to coordinate the cam generation (*startCamCalc*, *calcDone*, *calcError*, *calcBusy*).

Program sequence

The program sequence of the state machine of the function block (*FBTravCntrl*) is shown in the following diagram. Each bracket in the diagram describes a state.

Figure 5-2 State diagram



The input values and the user data are checked during the initialization. If no errors are identified when making the check, the state machine is started and the sequence shown executed. If an error occurs, the state machine is brought into the error state and the traverser (traversing arm) is immediately held. The error is always displayed at the *error* and *errorID* outputs. The block must then be re-started using a rising edge at the *enable* input and then traversing operation can be continued with the last direction of motion that was saved– as well as the saved layer counter. The layer counter can also be reset in operation using the corresponding inputs which will result also in a reset regarding all layer dependant coil parameters (not, if an external source is used to supply the diameter value).

If user data are changed in the traversing mode, then these are activated:

- In the next newly calculated traversing cycle – *parameterChangeMode* = *NEXT_NEW_CAM*
- In the next traversing cycle – *parameterChangeMode* = *NEXT_CAM*
- Immediately – *parameterChangeMode* = *IMMEDIATELY*

The user must ensure, that the newly calculated traversing profile (after parameter change or layer counter reset) overlaps with the active traversing profile. Otherwise the traversing will be stopped and error will be set.

A traversing cycle always starts at position A.

The traversing axis is stopped if the block is deactivated in operation.

Winding mode

The winding mode is defined using input parameter *windingMode*. Four operating modes are possible:

- Winder from the top
- Winder from the bottom
- Unwinder from the top
- Unwinder from the bottom

Table 5-27 Winding mode

Mode	Representation	Description
Rewind from above		<p>V: Machine velocity F: Tension in the material web</p> <p>n: Speed of the roll being wound n_o: Velocity override for speed controller override</p> <p>M_f: Frictional torque M_p: Precontrol torque M_T: Tension torque</p>
Rewind from below		
Unwind from below		
Unwind from above		

Note The operating mode is defined for a positive machine velocity. If the machine velocity is negative, then the winding behavior is inverted (for example, a winder from the top becomes an unwinder from the top when the web velocity is inverted).

Enable traverser after interruption of traversing cycle

To enable the traverser application after an interruption (*enable* = FALSE), especially if the profile parameter have changed, the following conditions should be considered. Especially the current position of the traverser should be monitored at a restart:

Table 5-28 Enable conditions

Switch on condition	Enable behaviour
After switch on the traverser should start at the same position (same layer) with the same profile parameters as before.	No initialization of the sTravActParam necessary <pre>eLTravEnableBehaviour := NO_INITIALIZATION; resetLayerCount := FALSE;</pre>
After switch on the traverser should start a new coil (Layer = 0) but with the same profile parameters. The current position of the traverser did not change.	No initialization of the sTravActParam necessary <pre>eLTravEnableBehaviour := NO_INITIALIZATION; resetLayerCount := TRUE;</pre>
The profile parameter have changed (e.g. r32StartPosA/B) when the traverser is disabled (enable = FALSE). The traverser should start at layer zero.	Initialization of the sTravActParam necessary <pre>eLTravEnableBehaviour := INITIALIZE_PROFILE; resetLayerCount := FALSE;</pre>
The profile parameter have changed (e.g. r32StartPosA/B) when the traverser is disabled (enable = FALSE). The traverser should start at the last saved layer.	Initialization of the sTravActParam necessary <pre>eLTravEnableBehaviour := INITIALIZE_PROFILE; resetLayerCount := FALSE;</pre> <p>Additionally set:</p> <ul style="list-style-type: none"> • Last active values (e.g. sTravActParam.r64ActPosA/B) as new set parameters (sTravSetParam.r64StartPosA/B) • Last diameter as new start diameter

NOTE The initialization depends on the configuration parameter eLTravEnableBehaviour. Is this value set to „INITIALZE_PROFIEL“ (default value), the active parameters (sTravActparam) will be initialized with a rising edge of the enabel input. Otherwise the last safed active parameters will be used for calculation.

5.5.3 Error Messages

Warnings

Warning messages are displayed using the state of the *error* and *errorID* output: *error* = FALSE and *errorID* <> 16#0000_0000.

Table 5-29 Warning messages

ErrorID	Description
16#0000_0000	No warning present
16#0401_0013	The layer counter has a negative value.
16#0401_00EE	The value of the external diameter less than the specified coil core.

Errors

Error messages are displayed using the state of the *error* and *errorID* output: *error* = TRUE and *errorID* <> 16#0000_0000.

Table 5-30 Error messages

ErrorID	Description
16#0000_0000	No error
16#0401_8001	Invalid TO reference at input travCamA. Check the interconnected reference.
16#0401_8002	Invalid TO reference at input travCamB. Check the interconnected reference.
16#0401_8003	Invalid TO reference at input winderAxis. Check the interconnected reference and the type (positioning axis or external encoder).
16#0401_8004	Invalid TO reference at input traverserAxis. Check the interconnected reference and the type (synchronous axis).
16#0401_8005	Invalid synchronous object. Check the axis configuration.
16#0401_8007	The traverser axis (traverserAxis) is not activated. Activate the axis.
16#0401_8008	The traverser axis (traverserAxis) is not at standstill. Stop the axis but keep the closed-loop control activated.
16#0401_8009	The <i>_setMaster</i> command was not processed error-free. Check the configuration of the synchronous relationship between the winder axis (winderAxis) and traverser axis (traverserAxis).
16#0401_800A	The first cam (travCamA) was not able to be reset. Ensure that no other program parts have access to the cam.
16#0401_800B	The second cam (travCamB) was not able to be reset. Ensure that no other program parts have access to the cam.
16#0401_800C	Internal program error
16#0401_800D	User data error. Check that the limit values are maintained as described in the type definitions.
16#0401_800E	Cam calculation error. Check the error output (<i>errorID</i>) of function block FBCamCalc.
16#0401_8010	When calling the <i>_getCamLeadingValue</i> function, an error occurred. Check the status of the cam object.
16#0401_8011	When calling the <i>_enableCamming</i> function, an error occurred. Check whether the cam is valid and whether the master axis is correctly configured.
16#0401_8012	Error when stopping the traverser axis.
16#0401_8013	No CAM is active by the time of manual reversal.

5 Functional description

ErrorID	Description
16#0401_8014	<p>At traversing start: the position of the traverser axis is located outside the coil edges. The coil edges are obtained from sTravSetParam.tolStartPos, sTravActParam.actPosA and sTravActParam.actPosB.</p> <p>Adapt the coil edges or set the starting position within the defined coil edges.</p> <p>During active traversing: after the reset of the traversing profile or parameter changes the newly calculated cam does not overlap the active cam in position ($posANEW > posBOLD$ or $posBNEW < posAOLD$).</p>
16#0401_8016	The TO reference is identical at the travCamA and travCamB inputs. Use two different cams.
16#0401_8017	Error when resetting the synchronous object
16#0401_8018	Error when scaling the cam (_setCamScale)
16#0401_8019	In the dynamic values (sStopDynamics; sSyncDynamics) is at least one value invalid. (velocity, acceleration, deceleration must be unequal 0)
16#0401_8020	The start position of traverser is not in segment 2 or 3 with conical coils.
16#0401_8FFF	Internal program error

5.6 FBCamCalc

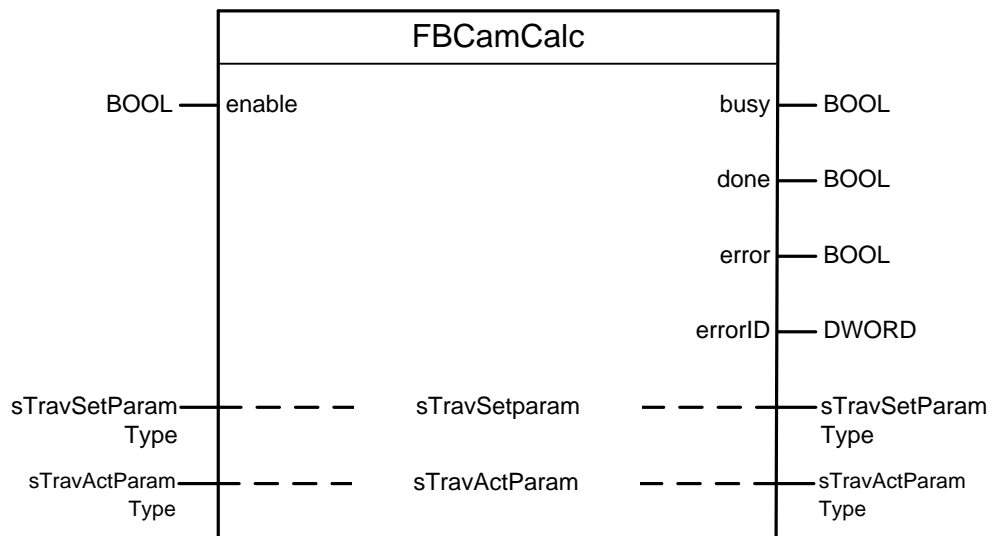
Task

This block calculates the motion profile of the traverser functionality and from this motion profile generates a cam.

The block must be called in a sequential task (motion task)

Schematic LAD Representation

Figure 5-3 FBCamCalc



© Siemens AG 2020. All rights reserved

5.6.1 Input and Output Parameters

Input parameters

Table 5-31 Input parameters

Name	Data type	Initial value	Description
enable	BOOL	FALSE	Rising edge: The block is activated and the cam calculation started Falling edge: The block and the outputs are reset.

Input/output parameters

Table 5-32 Input/output parameters

Name	Data type	Initial value	Description
sTravSetParam	sTravSetParam Type		Details on this structure are included in the data type description
sTravActParam	sTravActParam Type		Details on this structure are included in the data type description

Output parameters

Table 5-33 Output parameters

Name	Data type	Initial value	Description
done	BOOL	FALSE	TRUE: The cam calculation has been completed. Is used to coordinate with FBTravCntrl.
busy	BOOL	FALSE	TRUE: The cam calculation is active Is used to coordinate with FBTravCntrl.
error	BOOL	FALSE	TRUE: Error Is used to coordinate with FBTravCntrl.
errorId	DWORD	16#0000_0000	Error identification For warning messages, an error identification (error ID) is output, however, the error output is not set.

5.6.2 Functionality

The function block calculates the motion profile for the traverser axes – based on the specification entered - and uses this profile to generate a cam.

The function block transfers its status using the *busy*, *done* and *error* parameters. If the cam calculation is active, this is indicated at the *busy* output (*busy* = *TRUE*). If errors occur when checking, calculating or generating, the *error* output is set and generation interrupted. The *done* output is set if the cam has been successfully generated.

The calculated profile depends on the setting of the parameter *eTraversingMode* of the structure *sTravSetParam*.

Traversing mode CONTINUOUSLY

Four modes are available to calculate the motion profile. These modes can be selected in the *sTravSetParam* structure using the *eCalcMode* parameter.

ACCELERATION_ANGLE

The acceleration angle is adapted in this mode in order to reach the specified displacement angle. The acceleration angle may only move within the defined limits. If the calculation is not possible within these limits, processing is interrupted with an error.

WAITING_ANGLE

In this mode, the waiting angle is adapted to achieve the specified displacement angle. The waiting angle may only move within the defined limits. If the calculation is not possible within these limits, processing is interrupted with an error

WINDING_STEP

In this mode, the winding step is adapted to reach the specified displacement angle. The winding step may only move within the defined limits. If the calculation is not possible within these limits, processing is interrupted with an error.

SETPOINTS

In this mode, the setpoints are used for the waiting angle, the winding step and the acceleration angle. No adaptations are carried-out to reach the parameterized displacement angle. If the calculation is not possible, processing is interrupted with an error.

Traversing mode STEP_WIND

The profile in this mode will be calculated by the following parameters:

Table 5-34 Configuration parameters STEP_WIND

sTravSetParam	Description
r32StartPosA	[mm] Startpoint of the profile
r32LayserPerStep	[-] Number of layers per step
u16NumberOfSteps	[-] Number of steps
r32SetWaitAng	[°] Additionally waiting angle at each step
r32stepWindWaitAngle	[°] Waiting angle at Position A/B
r32SetAccAng	[°] Acceleration angle related to winder axis
r32SetWindStep	[mm/rev] Wind step as distance between the steps (traverser)

The profile starts in position A where the traverser is standing still while the winder is moving an angle defined by r32LayerPerStep multiplied by 360° and additionally the r32SetWaitAng. To respect the motion of the winder while the traverser is moving to the next step, the half of the acceleration angle is also used in this calculation. For the transition a polynom 5th order is used.

5.6.3 Error Messages

Warnings

Warning messages are displayed using the state of the *error* and *errorID* output: *error* = FALSE and *errorID* <> 16#0000_0000.

Table 5-35 Warning messages

ErrorID	Description
16#0000_0000	No warning
16#0402_0002	The layer counter contains a negative value. If winding operation is active, reset the layer counter - if unwinding operation is active, set the layer counter to a value that corresponds to the number of layers on the coil.
16#0402_000C	The value of the external diameter is less than the specified coil core.

Errors

Error messages are displayed using the state of the *error* and *errorID* output: *error* = TRUE and *errorID* <> 16#0000_0000.

Table 5-36 Error messages

ErrorID	Description
16#0000_0000	No error
16#0402_8001	The sTravActParam.selectedCam variable does not contain a valid reference for a cam. Check the validity of the value
16#0402_8003	User data error. Check the validity of the values – specifically regarding the value ranges.
16#0402_8004	Invalid coil edge points. Check the selected coil profile.
16#0402_8005	It is not possible to calculate the motion profile with the current settings. The calculated acceleration angle or the waiting angle lies outside the parameterized limits. The return value of the FCChkCalc function provides detailed information.
16#0402_8006	It is not possible to calculate the motion profile with the current settings. Spikes 1 and/or 2 are too high or too long, or the winding step is too high. Correspondingly change the parameters.
16#0402_8007	It is not possible to calculate the motion profile with the current settings. Spikes 3 and/or 4 are too high or too long, or the winding step is too high. Correspondingly change the parameters.
16#0402_8008	Error when resetting the cam. Ensure that the cam is not being used elsewhere.
16#0402_8009	Error when writing to the cam (_addSegmentToCam). Ensure that the cam is not being used elsewhere.
16#0402_800A	Error when writing to the cam (_addPointToCam). Ensure that the cam is not being used elsewhere.
16#0402_800B	Error when writing to the cam (_interpolateCam). Ensure that the cam is not being used elsewhere.
16#0402_800D	User data error. Check that the limit values are maintained as described in the type definitions for sTravSetParam.

5.7 FCChkCalc

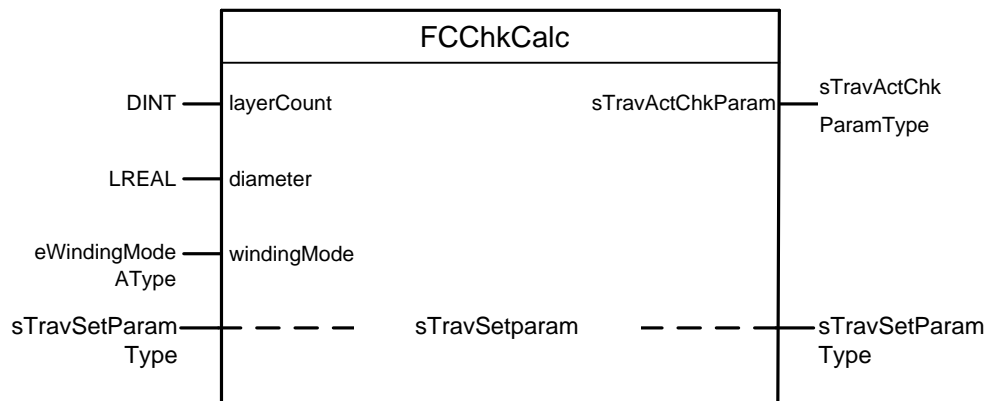
Task

The block performs a plausibility check for the specified user parameters and generates an array of messages that can be evaluated.

The block can be called in any task.

Schematic LAD Representation

Figure 5-4 FCChkCalc



5.7.1 Input and Output Parameters

Even if the variable type used here is the same as at *FBTravCntrl* or at *FBCamCalc*, another instance should be used to make the check, otherwise no testing is possible without influencing the traversing.

Input parameters

Table 5-37 Input parameters

Name	Data type	Initial value	Description
layerCount	DINT	0	Number of layers to be checked This value describes the number of layers that are checked for the specified traverser parameters. This check is important as several traverser parameters change dependent on the layer.
diameter	LREAL	100.0	[mm] If an external diameter is used to adapt the traverser parameters, then it must also be assigned to this parameter. The value must be positive.

Name	Data type	Initial value	Description
windingMode	eWindingMode AType	REWIND_FROM_ ABOVE	Winding mode: UNWIND_FROM_ABO VE: Unwinder from the top UNWIND_FROM_BEL OW: Unwinder from the bottom REWIND_FROM_ABO VE: Winder from the top REWIND_FROM_BELO W: Winder from the bottom

Input/output parameters

Table 5-38 Input/output parameters

Name	Data type	Initial value	Description
sTravSetParam	sTravSetParam Type		Details on this structure are included in the data type description

Output parameters

Table 5-39 Output parameters

Name	Data type	Initial value	Description
sTravActChkParam	sTravActChk ParamType		Details on this structure are included in the data type description

5.7.2 Functionality

This function performs a plausibility check for the user parameters that are defined in the structure *sTravSetParam*. The check includes the comparison of the user parameters with the basic parameter limits – as they are described in the type declaration. If limit value violations are identified, the corresponding error or warning signal is set as well as the group error or group warning bit.

In addition, the traversing profile of the traverser axis is calculated and checked. The calculation is always based on the actual layer counter or diameter (depending on the configuration, external diameter or layer counter).

- If the external diameter is used, then the check is carried-out based on the actual diameter. This means that the coil coordinates are not adapted for the check as no appropriate values are available for the simulation. The external diameter will be rounded internally. Using the core diameter and the thickness parameters an internal virtual layer count is created.
- If the traverser (layer counter) calculates the diameter, the check is performed – including the coil coordinates and adaptations.

5.7.3 Error Messages

The entries of the error data field are shown in the following table. The corresponding bit is set if an error is shown.

Table 5-40 Error array

Bit	Type	Description
00	WARNING	The layer counter is negative.
01	ERROR	The absolute value of the coil profile angle of the lefthand coil end point is greater than or equal to 90 degrees in mode 1.
02	ERROR	The absolute value of the coil profile angle of the righthand coil end point is greater than or equal to 90 degrees in mode 1.
03	ERROR	The material thickness is invalid.
05	ERROR	The coordinates of the coil end point are invalid.
06	ERROR	The coil core diameter is invalid
07	ERROR	The stroke is negative.
08	ERROR	The spike velocity/position of spike 1 is invalid.
09	ERROR	The spike length of spike 1 is negative.
10	ERROR	The spike velocity/position of spike 2 is invalid.
11	ERROR	The spike length of spike 2 is negative.
12	ERROR	The spike velocity/position of spike 3 is invalid.
13	ERROR	The spike length of spike 3 is negative.
14	ERROR	The spike velocity/position of spike 4 is invalid.
15	ERROR	The spike length of spike 4 is negative.
16	ERROR	The limits of the acceleration angle are invalid.
17	ERROR	The limits of the waiting angle are invalid.
18	ERROR	The limits of the winding step are invalid.
19	ERROR	The setpoint for the waiting angle is negative or the ratio for waiting angle distribution is invalid.
20	ERROR	The setpoint for the winding step is negative or zero.
21	ERROR	The setpoint for the displacement angle is negative.
22	ERROR	The setpoint for the acceleration angle is negative or zero.
23	WARNING	One of the two parameters from spike 1 is zero
24	WARNING	One of the two parameters from spike 2 is zero
25	WARNING	One of the two parameters from spike 3 is zero
26	WARNING	One of the two parameters from spike 4 is zero
27	ERROR	The value of the layer counter is too high, the calculated coordinate of the lefthand coil edge is greater than the righthand coordinate.
28	ERROR	The interval defined by the parameterized limit value does not allow a solution to be found in the actual calculation mode.
29	ERROR	The length of the phase for the constant velocity in the first section of the traversing cycle (FCA) is negative; either spikes 1 and/or 2 is/are too long and/or have an excessively high velocity setpoint. The winding step and the acceleration angle are too large.

5 Functional description

Bit	Type	Description
30	ERROR	The length of the phase for the constant velocity in the first section of the traversing cycle (BCA) is negative; either spikes 3 and/or 4 is/are too long and/or have an excessively high velocity setpoint. The winding step and the acceleration angle are too large.
31	ERROR	The winding step or a spike is zero.
33	WARNING	The spike adaptation mode 2 is selected, however, a correct adaptation factor was not selected.
34	WARNING	The actual diameter is not valid (it is less than the coil core diameter)
35	ERROR	Error when calculating the diameter-dependent profiles. The layer counter must be positive if the diameter is internally calculated.
36	ERROR	Invalid winder mode
37	ERROR	Tolerance bandwidth of the start position adaptation for synchronizing is negative.
38	ERROR	Dynamic synchronization velocity = 0
39	ERROR	Dynamic synchronization acceleration = 0
40	ERROR	Dynamic synchronization deceleration = 0
41	ERROR	Dynamic stop velocity = 0
42	ERROR	Dynamic stop acceleration = 0
43	ERROR	Dynamic stop deceleration = 0
44	ERROR	Calculated layer angle <= 0.0
45	ERROR	Number of steps <= 0
46	ERROR	Number of steps > LTRAVLIB_MAX_NUMBER_OF_STEPS
47	ERROR	Maximum number of iterations reached (LTRAVLIB_MAX_LOOP_COUNT) → No solution possible for entered parameters

5.8 FBCalcPrior

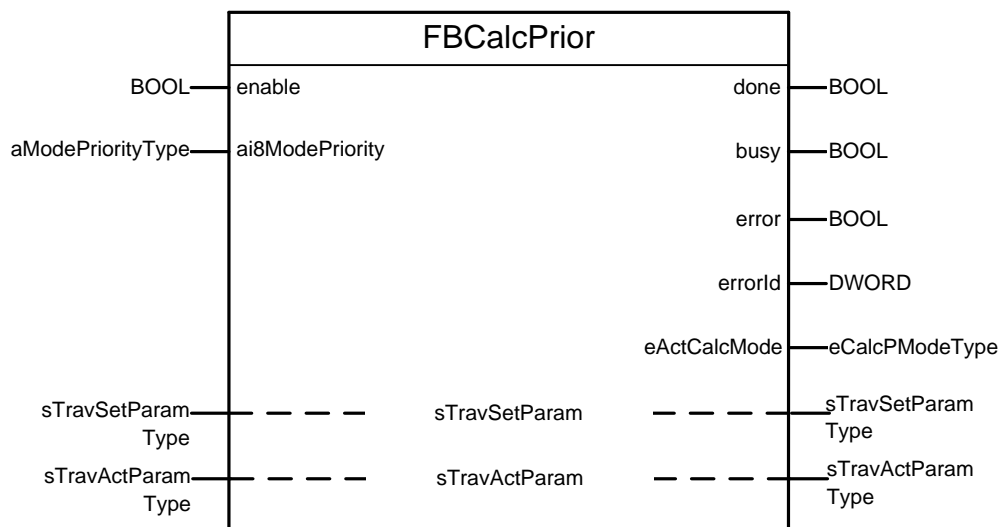
Task

This block supports the priority-based traversing profile calculation and is used as expansion for the *FBCamCalc* that is internally called in this block. To use this, the *FBCamCalc* call in the motion task is replaced by this block; however, the coupling to the *FBTravCntrl* remains as for *FBCamCalc*.

The block must be called in a sequential task (motion task).

Schematic LAD Representation

Figure 5-5 FBCalcPrior



© Siemens AG 2020. All rights reserved

5.8.1 Input and Output Parameters

Input parameters

Table 5-41 Input parameters

Name	Data type	Initial value	Description
enable	BOOL	FALSE	TRUE: The block is activated FALSE: The block is deactivated and the outputs reset
ai8ModePriority	aModePriority Type		Array of calculation priorities. The details of this structure are described in the Chapter of the type definitions.

Input/output parameters

Table 5-42 Input/output parameters

Name	Data type	Initial value	Description
sTravSetParam	sTravSetParam Type		Details on this structure are included in the data type description
sTravActParam	sTravActParam Type		Details on this structure are included in the data type description

Output parameters

Table 5-43 Output parameters

Name	Data type	Initial value	Description
done	BOOL	FALSE	TRUE: The calculation has been completed
busy	BOOL	FALSE	FALSE: No calculation active TRUE: The calculation is active
error	BOOL	FALSE	FALSE: No error TRUE: An error occurred during the calculation
errorId	DWORD	16#0000_0000	Error identification For alarm messages, an error identification is output, but the error output is not set.
eActCalcMode	eCalcPMode Type	NA	Last calculation mode used

5.8.2 Functionality

The block can be used as an alternative to *FBCamCalc*. When compared to *FBCamCalc*, the calculation modes can be assigned priorities to calculate the traversing profile in this block. This means that if the traversing profile cannot be calculated with the calculation mode assigned the highest priority, then the calculation is performed with the calculation mode of the next highest priority. This procedure is repeated until either a solution is found or no other calculation mode is available. If this is the case, the calculation is interrupted with an error.

If an error is identified during the plausibility check of the user parameters, block processing is immediately interrupted without performing a calculation.

The function block is integrated in the execution system in the same way as the *FBCamCalc*. Interlinking with the *FBTravCtrl* via global variables is also the same. The only difference to *FBCamCalc* is the additional parameter for the array with the calculation priorities.

If the priority assignment of the calculation modes is changed, then the user must ensure that *FBTravCtrl* identifies this change and also accepts it. For instance, this can be realized by changing the calculation mode in the *sTravSetParam.eCalcMode* structure (the input is not used for the calculation). The limit values of the calculation modes being used must also be adapted where relevant.

Prioritization of the calculation modes

Input *ai8ModePriority* is available to assign priorities to the individual calculation modes. Every array element stands for the priority of the corresponding calculation mode. The following applies here:

- Priority ≤ 0 : The calculation mode is not used
- Priority > 0 : The calculation mode is used. The calculation mode with the highest number has the highest priority. Identical priorities for several modes is not permissible (with the exception of values ≤ 0 !).

An example with different priorities - as well as the assignment of the priorities - is shown in the following:

Fig. 5-6 Example of prioritizing the calculation modes

```
ai8Modepriority[1] := 7;      // Acceleration Angle
ai8Modepriority[2] := 0;      // Waiting angle
ai8Modepriority[3] := 14;     // Winding step
ai8Modepriority[4] := 0;      // Setpoints
```

Calculation modes WAITING_ANGLE and SETPOINTS are deactivated. The calculation is first made in the WINDING_STEP mode; if a solution is not obtained in this mode, then the calculation is repeated in the ACCELERATION_ANGLE mode.

Note

Prioritization can be deactivated if all priorities except for one are set to values less than zero. In this case, the principle of operation is identical to that of block [FBCamCalc](#).

5.8.3 Error Messages

Warnings

Warning messages are displayed using the state of the *error* and *errorID* output:
error = FALSE and errorID <> 16#0000_0000.

Table 5-44 Warning messages

ErrorID	Description
16#0000_0000	No warning
16#0404_0002	The layer counter has a negative value. Correct the value or restart the application.
16#0404_000C	The value of the external diameter is less than the coil core diameter.

Errors

Error messages are displayed using the state of the *error* and *errorID* output: *error* = TRUE and *errorID* <> 16#0000_0000.

Table 5-45 Error messages

ErrorID	Description
16#0000_0000	No error
16#0404_800C	Invalid priorities, the priority assignments of the modes must have different values.
16#0404_800D	There is no solution for the specified parameters and modes.
16#0404_8FFF	Internal program error
16#0404_8001	Invalid TO reference in the sTravActParam.selectedCam variable. Check whether the object was set-up.
16#0404_8003	Error in the user parameters. Check the user parameters to ensure that the limits specified in the Chapter of the type definitions are maintained. Detailed information can be obtained by evaluating FCChkCalc.
16#0404_8004	The coordinates of the coil end points are interchanged (position A > position B). Check the selected coil profile and the value of the layer counter.
16#0404_8008	Error when resetting the cam. Ensure that the cam is not being used by another program selection.
16#0404_8009	Error when generating the cam (_addSegmentToCam). Ensure that the cam is not being used by another program selection.
16#0404_800A	Error when generating the cam (_addPointToCam). Ensure that the cam is not being used by another program selection.
16#0404_800B	Error when generating the cam (_interpolateCam). Ensure that the cam is not being used by another program selection.

5.9 FBChkPrior

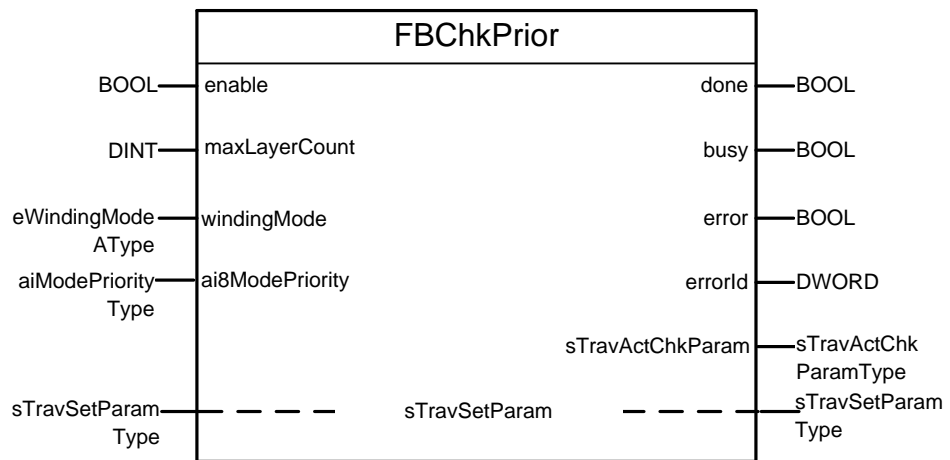
Task

The block performs the plausibility check of the specified user parameters and generates an array of error messages that can be evaluated. Contrary to *FCChkCalc*, the layer dependency is taken into account in this function.

The block must be called in a sequential task (motion task).

Schematic LAD Representation

Figure 5-7 FBChkPrior



5.9.1 Input and Output Parameters

Input parameters

Table 5-46 Input parameters

Name	Data type	Initial value	Description
enable	BOOL	FALSE	TRUE: The function is activated FALSE: The execution of the function is interrupted
maxLayerCount	DINT	0	Number of layers to be checked.
windingMode	eWindingMode AType	REWIND_FROM_ABOVE	Winding mode: UNWIND_FROM_ABOVE UNWIND_FROM_BELOW REWIND_FROM_ABOVE REWIND_FROM_BELOW
ai8ModePriority	aModePriority Type		Array of the calculation priorities. The details of this structure are described in the Chapter of the type definitions.

Input/output parameters

Table 5-47 Input/output parameters

Name	Data type	Initial value	Description
sTravSetParam	sTravSetParam Type		Details on this structure are included in the data type description

Output parameters

Table 5-48 Output parameters

Name	Data type	Initial value	Description
done	BOOL	FALSE	TRUE: The calculation has been completed
busy	BOOL	FALSE	FALSE: No calculation active TRUE: The calculation is active
error	BOOL	FALSE	FALSE: No error TRUE: An error occurred during the calculation
errorId	DWORD	16#0000_0000	Error identification For warning messages, an error identification is output - however the error output is not set.
sTravActChkParam	sTravActChk ParamType		Details on this structure are included in the data type description

5.9.2 Functionality

This function block performs the plausibility check of the user parameters with priority assignment of the calculation mode for a defined layer range. For the plausibility check of the parameters, the *FCChkCalc* function is internally called for each layer. If errors are identified when making the check, the calculation mode is changed – corresponding to the parameterized priority assignment and the calculation repeated. If a solution is still not possible, processing is interrupted and the appropriate error message output. In so doing, the error bits of *FCChkCalc* are transferred.

The simulation does not function with an external diameter as the diameter values then cannot be simulated.

5.9.3 Error Messages

Errors

Error messages are displayed using the state of the *error* and *errorID* output: *error* = TRUE and *errorID* <> 16#0000_0000.

Table 5-49 Error messages

ErrorID	Description
16#0000_0000	No error
16#0405_800C	Invalid priorities: The priority assignment of the modes must have different values.
16#0405_800D	There is no solution for the specified parameters and modes.
16#0405_800E	The external diameter calculation is not supported.
16#0405_800F	Calculation error, check the error array
16#0405_8010	Negative value of the layer counter limit
16#0405_8FFF	Internal program error

6 FAQ

6.1 Layer dependant adoptions

With the Traversing application it is possible to use functions that change the profile during production, like coil profiles or adaptations. If such functions are used it is strongly advised to check the plausibility of the parameters in advance.

For coil profiles or adaption there is a new cam calculated in every traversing cycle. It is possible, that suddenly after numerous cycles the profile calculations will lead to a calculation error, for example the coil endpoints invert or the tolerance windows for the parameters (acceleration angle, winding step, waiting angle) can no longer be obtained.

To avoid this the traversing parameters need to be checked for validity after input or any parameter change. For this plausibility check we offer the unit *FBChkPrior()* which can verify the parameters for a defined number of layers. With this functionality it is possible to verify the complete rewinding process at parameter input. The number of layers for the plausibility check can be derived from the shut-off criteria (layer count, material length, diameter) of the rewinder.

To successfully implement this functionality you must define a separate data structure for the input parameters which can independently used for the plausibility check without disturbing the actual traversing process. The parameters can be transferred to the traversing block after successfully passing *FBChkPrior()*, if the check is unsuccessful the traversing should continue with the unchanged, old data set.

6.2 Winder axis diameter calculation

Opposite to winding without traversing there are some disturbing effects: the dancer position or the actual tension will change as the traversing arm reverses, since we have an abrupt layer jump. This disturbs the actual speed of the winder and so the diameter calculation. The diameter value needs to be stabilized using an appropriate method.

The *SIMOTION Winder* standard application offers numerous diameter calculation methods. For winders with traversed materials the best results are obtained with *DIAM_CALC_INTEGRAL* and *DIAM_CALC_POSITION*.

6.3 Impact on winder speed

As the traversing arm back and forth moves changes the path length of the material, increases and decreases. This effect is leveled out by the dancer.

To avoid or minimize this effect the setpoint velocity of the traverser axis can be used as a precontrol value added on top of the line speed for the winder axis. In the precontrol the geometric configuration of the material path must be respected.

The *FBWinder* block is able to accept the precontrol value if the line speed is transferred using the winder input structure "*lineAxisMotionVector*" and not with a technology object. In this case the precontrol value needs to be added to the above mentioned variable.

7 Collection of formulas

7.1 Coil profile definition formulas

The coil end positions – without stroke – are calculated using the following formulas. The result is always interpreted in [mm].

Coil profile mode, coil edge angle

$$posA_{act} = posA - \tan(coilAngA) * 2 * materialThickness$$

$$posB_{act} = posB - \tan(coilAngB) * 2 * materialThickness$$

If an external diameter value is used then instead of $2 * material\ thickness$ (above) the half of the diameter change is used for the adaption.

posA	[mm] calculated lefthand coil edge
posB	[mm] calculated righthand coil edge
startposA	[mm] coordinate of the lefthand coil edge, user parameter
startposB	[mm] coordinate of the righthand coil edge, user parameter
coilAngA	[°] material angle at the lefthand coil edge
coilAngB	[°] material angle at the righthand coil edge
Material thickness	[mm] effective material thickness

Coil profile mode, layer offset

$$posA_{act} = posA - \frac{2 * coilAngA}{1000}$$

$$posB_{act} = posB + \frac{2 * coilAngB}{1000}$$

posA	[mm] calculated lefthand coil edge
posB	[mm] calculated righthand coil edge
startposA	[mm] coordinate of the lefthand coil edge, user parameters
startposB	[mm] coordinate of the righthand coil edge, user parameters
coilAngA	[μm] layer offset at the lefthand coil edge
coilAngB	[μm] layer offset at the righthand coil edge

Parameter changes, recursivity

Changed coil parameters are effective from the next cycle. The profile must be reset explicitly with the reset layer counter input. If external diameter is used the profile is reset with respect to the actual value that is shown at actual diameter. The internally calculated diameter is reset with the layer counter immediately.

7.2 Diameter calculation

The actual diameter is calculated using the following formula. The result is always interpreted in mm.

$$actual\ diameter = core\ diameter + layer\ count * 2 * material\ thickness$$

Core diameter	[mm] Coil core diameter
Layer count	[-] layer counter
Material thickness	[mm] effective material thickness

The effective material thickness is only calculated if the material width parameter is defined.

$$effective\ thickness = material\ thickness * \frac{material\ width}{winding\ step}$$

Material width	[mm] material width
Winding step	[mm/rev] winding step In calculation mode WINDING_STEP the minimal winding step is used otherwise the setpoint winding step is used.
Material thickness	[mm] material thickness
Effective thickness	[mm] effective material thickness

7.3 External diameter rounding

If external diameter is used the actual value will be internally rounded using the following formula. The floor function is used to round to the largest previous integer.

$$diameter_{int} = diameter_{core} + material\ thickness * \frac{diameter_{ext} - diameter_{core}}{material\ thickness}$$

diameter _{INT}	[mm] internal, rounded diameter value
diameter _{CORE}	[mm] core diameter
diameter _{EXT}	[mm] external diameter value
material thickness	[mm] effective material thickness

7.4 Calculating the maximum length of the cam

The maximum length of the traversing cycle is estimated using the following formula.

$$cycle\ length_{max} \approx \frac{720 * TW_{max}}{WS_{min}} + 2 * AA$$

TW_{max}	[mm] maximum coil width (position B – position A)
WS_{min}	[mm] minimum winding step
AA_{max}	[°] maximum acceleration angle

7.5 Adapting the stroke and the spike in the additional velocity mode

The stroke and the spikes additional velocity is adapted using a factor. This factor is called the *layer factor*. The layer factor initially at profile initialization is set to the quotient of the actual and the core diameter (with internal diameter calculation this equals 1.0).

The adaption is realized based on the following formula:

$$parameter_{act} = \frac{parameter_{initial}}{layer\ factor}$$

The *parameter* is the stroke or the spike velocity. The layer factor is recalculated in every traversing cycle as follows:

$$layer\ factor = layer\ factor + \frac{4 * material\ thickness}{core\ diameter}$$

In case the use of external diameter instead of using $4 * material\ thickness$ the change in the diameter value is used for the adaption.

Parameter	[mm] Stroke or [mm/rev] Additional velocity
Layer factor	[-] layer factor for adaption
Material thickness	[mm] effective material thickness
Core diameter	[mm] core diameter

7.6 Adaptation of the spike according the position offset mode

The value to adapt the spike according to the position offset mode is directly entered as offset per layer. The spike is calculated using the following formula:

$$parameter_{(layer=x)} = parameter_{(layer=0)} - x * correction_factor$$

x	[-] value of the layer counter
parameter(layer=x)	[mm] adapted position offset
parameter(layer=0)	[mm] user parameter, position offset for diameter = coil diameter
correction_factor	[mm] displacement / layer

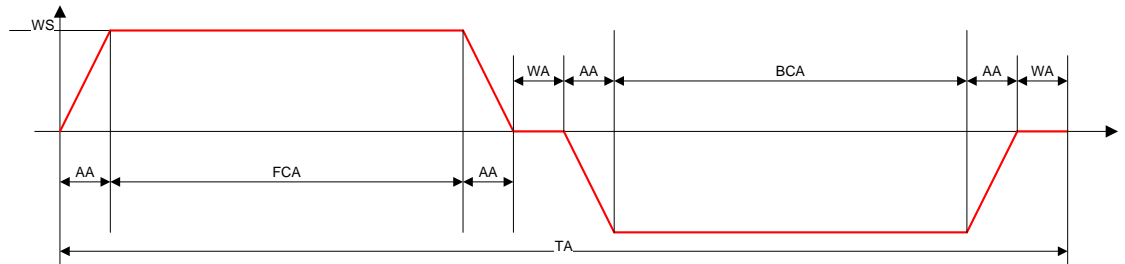
The spike is deactivated if the value of the layer counter is too high and therefore the resulting layer offset is negative. The spike is not adapted if the correction factor is negative.

7.7 Calculating the motion profile

Without spikes

The velocity profile of the traverser without spikes is shown in the following:

Figure 7-1 Velocity profile of the traverse without spikes



AA	[°] acceleration angle
WA	[°] waiting angle
WS	[mm/rev] winding step
FCA	[°] constant velocity phase when the traverser is moving forwards
BCA	[°] constant velocity phase when the traverser is moving backwards.
TA	[°] Total angle, cycle length

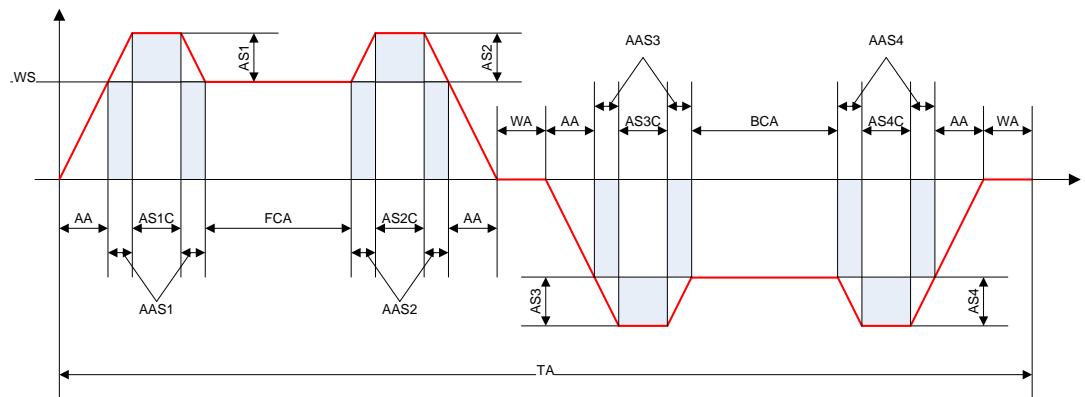
The following equations define the motion profile of a traversing cycle. TA [°] designates the length of the complete cycle as angle of the winder axis:

- $TA = 4 * AA + 2 * WA + FCA + BCA$
- $360 * TW = [AA * WS / 2] + [FCA * WS] + [AA * WS / 2]$
- $360 * TW = [AA * WS / 2] + [BCA * WS] + [AA * WS / 2]$

With spikes according to the additional velocity mode

The motion profile of the traverser with spikes in the additional velocity mode is shown in the following diagram:

Figure 7-2 Velocity profile of the traverser with spikes in the additional velocity mode



AS#	[mm/rev] additional velocity of the spike
AS#C	[°] spike length (constant velocity)
AAS#	[°] the acceleration angle of the spike to reach the additional velocity of the spike is calculated from the values of AA, WS and AS#.
AS	[mm/rev] sum of the additional velocities AS#
ASC	[°] sum of the spike lengths AS#C

The following equations define the motion profile of the traverser:

$$1. \quad TA = 4 * AA + 2 * WA + FCA + BCA + 2 * AAS1 + AS1C + 2 * AAS2 + AS2C + 2 * AAS3 + AS3C + 2 * AAS4 + AS4C$$

After simplification:

$$1. \quad TA = 4 * AA + 2 * WA + FCA + BCA + ASC + 2 * AA * AS / WS$$

Equation 1 calculates the complete angle. The displacement angle can be calculated with a modulo division with 360°.

The following equations form the integral over the velocity profile and therefore define the position profile. In [Figure 7-2](#), the area under the curve defines the position of the traverser. The forwards motion of the traverser ends at the end of the first waiting angle, the backwards motion at the end of the second waiting angle. The total surface defines the complete traversing path.

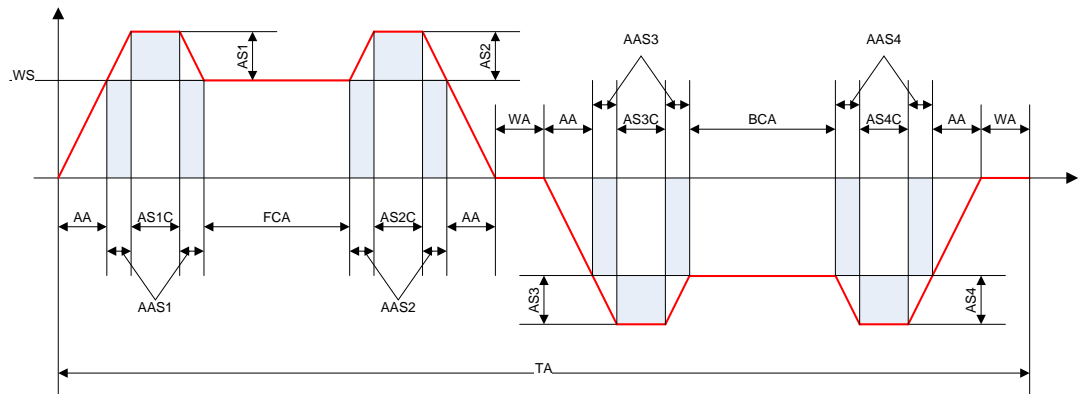
$$2. \quad 360 * TW = [AA * WS / 2] + [(AS1 / WS * AA) * WS] + [(AS1 / WS * AA) * AS1/2] + [AS1C * WS] + [AS1C * AS1] + [(AS1 / WS * AA) * WS] + [(AS1 / WS * AA) * AS1/2] + [FCA * WS] + [(AS2 / WS * AA) * WS] + [(AS2 / WS * AA) * AS2/2] + [AS2C * WS] + [AS2C * AS2] + [(AS2 / WS * AA) * WS] + [(AS2 / WS * AA) * AS2/2] + [AA * WS / 2]$$

$$\begin{aligned}
 3. \quad 360 * TW = & \quad [AA * WS / 2] \\
 & + [(AS3 / WS * AA) * WS] + [(AS3 / WS * AA) * AS3/2] \\
 & + [AS3C * WS] + [AS3C * AS3] \\
 & + [(AS3 / WS * AA) * WS] + [(AS3 / WS * AA) * AS3/2] \\
 & + [BCA * WS] \\
 & + [(AS4 / WS * AA) * WS] + [(AS4 / WS * AA) * AS4/2] \\
 & + [AS4C * WS] + [AS4C * AS4] \\
 & + [(AS4 / WS * AA) * WS] + [(AS4 / WS * AA) * AS4/2] \\
 & + [AA * WS / 2]
 \end{aligned}$$

With spikes according to the position offset mode

The motion profile of the traverser with spikes in the position offset mode is shown in the following diagram:

Figure 7-3 Velocity profile of the traverser with spikes in the position offset mode



AS# [mm] position offset of the spike#.
 AS [mm] sum of all position offsets

The following equations define the motion profile of the traverser:

$$1. \quad TA = 4 * AA + 2 * WA + FCA + BCA + AS1C + AS2C + AS3C + AS4C$$

After simplification:

$$1. \quad TA = 4 * AA + 2 * WA + FCA + BCA + ASC$$

Equations 2 and 3 are obtained as described for mode 1:

$$\begin{aligned}
 2. \quad 360 * TW = & \quad [AA * WS / 2] \\
 & + [AS1C * WS] + [AS1] \\
 & + [FCA * WS] \\
 & + [AS2C * WS] + [AS2] \\
 & + [AA * WS / 2]
 \end{aligned}$$

$$\begin{aligned}
 3. \quad 360 * TW = & \quad [AA * WS / 2] \\
 & + [AS3C * WS] + [AS3] \\
 & + [BCA * WS] \\
 & + [AS4C * WS] + [AS4] \\
 & + [AA * WS / 2]
 \end{aligned}$$

7.8 Abbreviations

Table 7-1 Abbreviations

Name	Description
AA	Acceleration angle
AAS#	Additional acceleration angle for spike #.
AAS	AAS1 + AAS2 + AAS3 + AAS4
AS#	Over-velocity or layer offset of the spike#
AS	AS1 + AS2 + AS3 + AS4
AS#C	Spike#
ASC	AS1C + AS2C + AS3C + AS4C
BCA	Angle of the constant velocity phase for backwards motion of the traverser (Backward Constant Angle).
DA	Displacement angle
FCA	Angle of the constant velocity phase for forwards motion of the traverser (Forward constant angle).
TA	Total angle, traverser cycle length in master degrees.
TW	Coil width (distance between positions A and B)
WA	Waiting angle
WS	Winding step

8 Appendix

8.1 Service and Support

Industry Online Support

Do you have any questions or need assistance?

Siemens Industry Online Support offers round the clock access to our entire service and support know-how and portfolio.

The Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs, application examples and videos – all information is accessible with just a few mouse clicks at:

<https://support.industry.siemens.com>

Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts. You send queries to Technical Support via Web form:

www.siemens.com/industry/supportrequest

SITRAIN – Training for Industry

With our globally available training courses for our products and solutions, we help you achieve with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to:

www.siemens.com/sitrain

Service offer

Our range of services includes the following:

- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog:

<https://support.industry.siemens.com/cs/sc>

Industry Online Support app

You will receive optimum support wherever you are with the "Siemens Industry Online Support" app. The app is available for Apple iOS, Android and Windows Phone:

<https://support.industry.siemens.com/cs/ww/en/sc/2067>

8.2 Application Support

Siemens AG
 Digital Factory Division
 Factory Automation
 Production Machines
 DF FA PMA APC
 Fraunauracher Str. 80
 91056 Erlangen, Germany
 mailto: tech.team.motioncontrol@siemens.com

8.3 Links and Literature

Table 8-1

No.	Topic
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link to this entry page of this application example https://support.industry.siemens.com/cs/ww/en/view/36037374
\3\	Link to the SIMOTION winder and tension control https://support.industry.siemens.com/cs/de/en/view/35818687
\4\	Link to the SIMOTION/SIMATIC/SINAMICS Converting Toolbox https://support.industry.siemens.com/cs/de/en/view/109744606

8.4 Change documentation

Table 8-2

Version	Date	Modifications
V1.0.0	15.06.2005	Created
V2.0.0	12.04.2006	Changes required by Styleguide / Encoder as master / independent from IPO-cycle time / compatibility with SIMOTION V4.0 / run-time optimization
V3.0.0	01.09.2008	Transfer from gearing to camming. Extended functionality with new calculation modes, spike, coil profiles.
V3.1.0	29.07.2009	Bugfixes, functional extensions
V3.2.0	15.04.2010	Bugfixes, functional extensions. Not Released.
V3.10.0	21.07.2010	New documentation template. Revised profile calculation, bugfixes.
V3.11.0	14.09.2011	Bugfixes, runtime optimization
V3.12.0	22.11.2011	Bugfix, Data consistency in FBCalcPrior
V3.12.1	28.06.2012	Bugfixes
V3.12.2	03.07.2012	Bugfixes
V3.13.0	18.01.2013	Functions for step wind implemented Bugfixes

Version	Date	Modifications
V3.13.1	28.03.2013	Bugfixes
V3.13.2	18.04.2013	Bugfixes
V3.14.0	29.08.2013	Bugfixes
V4.0.0	22.11.2013	Bugfixes, new nomenclature for data types and structures regarding styleguide, new library structure
V4.0.1	13.03.2014	Bugfixes
V4.0.2	14.04.2014	Bugfixes
V4.0.3	29.07.2014	Bugfixes
V4.0.4	17.03.2015	functional extension
V4.0.5	20.05.2016	Bug fixes, functional extension
V4.0.6	09/2018	Structural adaptations, Adaption to new example project
V4.0.7	09/2019	Added chapter on winding mode Added further description on the example HMI
V4.0.8	02/2020	No changes in documentation Update library LTravLib