



OMAC Packaging Working Group

PackSoft™ Committee

Technical Paper Packaging Application Function Block Library PackAL

Version 1.0

DISCLAIMER OF WARRANTIES

THIS DOCUMENT IS PROVIDED ON AN “AS IS” BASIS AND MAY BE SUBJECT TO FUTURE ADDITIONS, MODIFICATIONS, OR CORRECTIONS. OMAC HEREBY DISCLAIMS ALL WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, FOR THIS DOCUMENT. IN NO EVENT WILL OMAC BE RESPONSIBLE FOR ANY LOSS OR DAMAGE ARISING OUT OR RESULTING FROM ANY DEFECT, ERROR OR OMISSION IN THIS DOCUMENT OR FROM ANYONE’S USE OF OR RELIANCE ON THIS DOCUMENT.

Function blocks for Packaging Control

The following paper is a document developed within the OMAC PWG PackSoft committee. It summarizes the results of the OMAC PWG PackSoft committee work. The present specification is written thanks to contributions of its members:

Bernd Junker	Rovema
Bill Faber	Yaskawa
Bill Krah	ARC
Bob Kollmeyer	Giddings & Lewis
Bob Veltman	Tevopharm
Brad Weinschenker	Bosch Rexroth
Chuck Padvorac	Beckhoff
Curtis Basore	EJ Gallo
Daniel J. Throne	Bosch Rexroth
David Humphrey	ARC
David Newcorn	PackWorld Magazine
Dennis Daniels	ARC
Eelco van der Wal	PLCopen
Evan Hand	Kraft Food
Gerd Hoppe	Beckhoff
Istvan Ulvros	Tetrapack
Jay Clark	Roy g biv
Jeffrey A. Bock	Bosch Rexroth
Jeffry Wilkins	FMS Athens
Jerry Yen	GM
Joe Kimbrell	Automation Intelligence
John Kowal	Elau
Alfred Moeltner	Elau
John Michaloski	NIST
John Ryan	Siemens Energy & Automation
John Wenzler	Bosch Rexroth
Keith Spiro	Siemens Energy & Automation
Kenneth J. Ryan	Alexandria Technical College
Patrik Hug	Elau
Richard Covarrubia	Modicon
Sentekin Can	Purina
Shiva Sastry	University of Akron
Steve Ford	Mars
Susan Dorscheid	Giddings & Lewis
Terry Gustafson	LA Center of Technology
Thomas Cord	Elau
Tom Sasada	Yaskawa
Dan Whisel	Delta Systems
Dan Cadigan	BP Castrol
Jeff Brown	Beckhoff
Graham Harris	Beckhoff
Joseph Eiden	Schneider Electric
Kevin Hamilton	Schneider Electric
Keith Campbell	LA Center of Technology
Rudy Belliardi	Schneider Electric
Joe Faust	Douglas Machinery
Tom Jensen	Elau
Dough Imes	Hartness
Neville Cork	Hardness
Marc Bertrand	Stonetek
E. Airgoo	RA Jones
Tommy A. Pool	Kliklok Woodman
Scott M. Richards	Polytron
Mike Pieper	Siemens Energy & Automation
N'kosi Tyehimba	Schneider Electric
Walther Schuller	Baumüller

Change Status List:

Version number	Date	Change Comment
V 0.1	January 12 2004	Preliminary version, Input to Orlando Spring 2004 meeting, RFC
V 0.3	November 5, 2004	Preliminary version, RFC to Fall PackExpo 2004 meeting, RFC
V 0.7	February 2, 2005	Preliminary version, Input to Orlando Spring 2005 Meeting, internal distribution, RFC,
V 0.8	February 5, 2005	Preliminary version, Input from Orlando Spring 2005 Meeting to group, internal distribution, RFC
V 0.99	February 17, 2005	Preliminary version, Input from group, corrections to the v0.8, internal distribution, Request for Comment
V 1.0	March 29, 2005	Final version 1.0, Distribution

Contents

1. GENERAL	5
1.1. SCOPE.....	5
1.2. OBJECTIVES.....	5
1.2.1. General Objectives	6
1.2.2. Language context goals.....	6
1.2.3. Technical Units.....	6
1.2.4. Naming Conventions.....	6
1.2.5. Axis_REF Data type	7
1.2.6. Documentation Conventions.....	7
1.2.7. Compliance and Portability.....	7
1.3. CERTIFICATION.....	8
2. OVERVIEW OF THE DEFINED PACKAGING APPLICATION FUNCTION BLOCKS	8
3. PACKAGING APPLICATION FUNCTION BLOCKS.....	9
3.1. WIND / UNWIND AXIS (CONSTANT SURFACE VELOCITY, CSV MODE)	9
3.2. WIND / UNWIND AXIS (CONSTANT TORQUE, CT MODE).....	10
3.3. DANCER CONTROL	12
3.4. REGISTRATION	14
3.5. REGISTRATION_CORRECTION.....	15
3.6. INDEXING	18
3.7. BATCH COUNTER.....	19
3.8. DIGITAL PLS (PHASE LIMIT SWITCH OR CAM SWITCH)	20
3.9. OVERRIDE	23
3.10. JOGAXIS	25
3.11. FLYING SYNC	26
3.12. GEAR_IN WITH DYNAMIC GEAR FACTOR.....	29
3.13. MOTION COMMAND STOP WITH ORIENTED HALT	30
4. PACKAGING MACHINE STATE BLOCKS	32
4.1. GENERIC MACHINE STATE FUNCTION BLOCK	32
4.2. PS_PACKML_STATE_MACHINE.....	32
5. PACKAGING COMMUNICATION FUNCTION BLOCKS.....	35
5.1. POINT TO POINT COMMUNICATION BY FUNCTION BLOCK	35
PS_Send, PS_Receive.....	35
5.1.1. PS_Send.....	35
5.1.2. PS_Receive	36
5.2. MESSAGE OR TELEGRAM BASED COMMUNICATION.....	37
5.2.1. PS_CommRequest.....	37
5.2.2. PS_Indicate.....	38
5.2.3. PS_CommRespond.....	39
5.2.4. PS_CommConfirm.....	39
5.2.5. PS_CommNotify.....	40
5.2.6. PS_CommIndicate	41
5.3. COMMUNICATION BY PUBLISHER/SUBSCRIBER	42
5.3.1. PS_Publish.....	42
5.3.2. PS_Subscribe.....	43
GLOSSARY, REFERENCES AND STANDARDS	44
APPENDIX A. COMPLIANCE DECLARATION AND COMPLIANCE LIST.....	45
APPENDIX B: THE OMAC PWG LOGO AND IT'S USAGE.....	49

1. General

1.1. Scope

OMAC derives common solutions collectively for both technical and non-technical issues in the development, implementation, and commercialization of open architecture control technologies to maximize the business value of packaging machinery by developing guidelines that lead to the most appropriate application of advanced automation technology.

The PackSoft Subcommittee recommends that Technology Providers, OEM's and End Users adopt the IEC 61131-3 Standard for Programmable Controllers: Programming Languages for packaging machine automation, including motion control. The IEC 61131-3 standard applies to a wide range of programmable controllers and is commercially available in products from a number of Technology Providers. Use of the standard encourages software design that is hierarchical in nature, re-usable through program organization units; and the standard provides facilities for real-time exchange of information via communication networks.

The PackSoft Subcommittee recommends the use of PLCopen Function Blocks for Motion Control V1.0, the Extensions (Part 2) to this standard, and other coming PLCopen publications as they might be a fit, all based on IEC 61131-3.

The PackSoft Subcommittee develops Packaging Industry related Control Software guidelines that allow to commonly implement as technology, program machinery equipment and controls, maintain, train and learn the use of software on packaging industry devices.

1.2. Objectives

The objectives of the OMAC PWG PackSoft committee is to

- define software modules (based on appropriate standards, e.g. IEC61131-3) which describe basic packaging machinery control elements,
- develop a programming convention and a set of functional software elements which lend themselves to be become common use throughout the Packaging Industry to simplify the structure, understanding and handling of generic machine elements and their representation in software.
- define a set of functions which will serve the majority of packaging user's applications and needs
- promote the adoption of useful existing and emerging standards to own definitions.

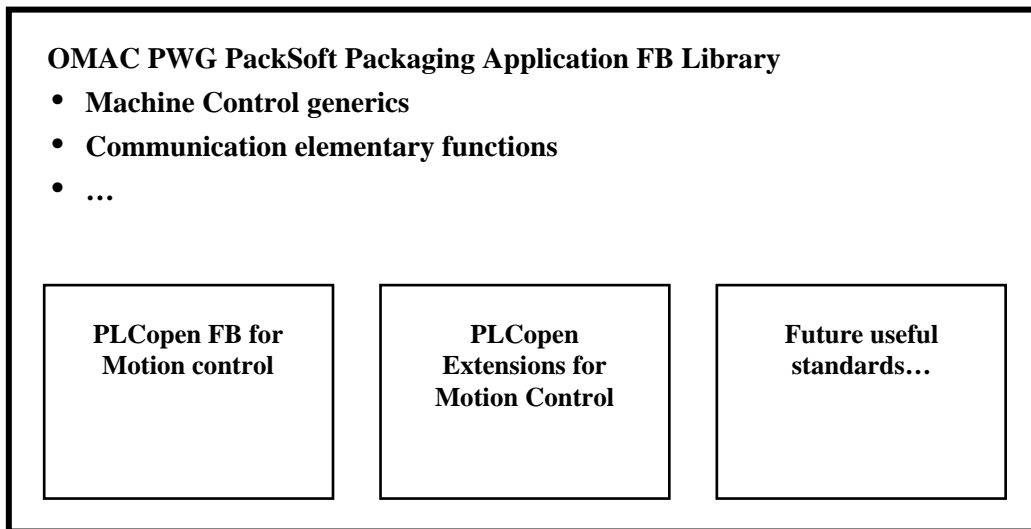


Fig. 1: Scope of definitions, embracing existing and emerging standards

1.2.1. General Objectives

- Machine related as first priority, process related as second
- Providing an easy-to-use interface to the Packaging Functionality
- Related to existing packaging standards
- Re-usable parts, usable in a wide range of applications.
- Application program should be implementable on any platforms
- Accepted / User-tested Functionality / Concepts providing the basis for FBs
- Providing a common basis, terminology, references
- Providing a ‘style guide’ for additional / future FBs
- Providing user guidelines / examples
- Standard Function Blocks Library for standard Packaging related Functionality
- Combining these FB’s to an application program needs an environment that is suitable for Packaging related applications. Requirements and restrictions for such an environment are partly dealt with in this standard.

1.2.2. Language context goals

- Focus on definition of
 - Function block interfaces and behavior
 - Data Typesaccording to the IEC 61131-3.
- These FBs and data types can be used in all IEC61131-3 languages.
- The examples in this draft are just given informatively in textual (IL, ST) and graphical (LD, FBD, SFC) IEC61131-3 languages.
- The contents of the function blocks can be implemented in any programming language (e.g. IEC61131-3 ST, C) or even in firmware or hardware. Therefore the content should not be expected to be portable.
- Reusable applications composed from these Function Blocks and data types can be achieved by PLCopen Conformity Level and Reusability Level if IEC61131-3 languages and future PLCopen certification and exchange standards.
- This specification shall be seen as an open framework without hardware dependencies. It provides openness in the implementation on different platforms such as fully integrated, centralized or distributed systems. The actual implementation if the Function Blocks themselves is out of the scope of this Packaging FB Library.

1.2.3. Technical Units

The only specification is made on the length unit (noted as [u]) that is to be coherent with its derivatives i.e. (velocity [u/s]; acceleration [u/s²]; jerk [u/s³]). Nevertheless, the unit [u] is not specified (manufacturer dependent). Only its relations with others is specified.

1.2.4. Naming Conventions

All naming conventions shall adhere to common rules developed within OMAC PWG. All standardization conventions, e.g. for definition of technical values, general coding rules, etc. shall adhere to the conventions developed by PLCopen Motion Task Force as described in the Function Blocks for Motion Control V1.0 standard.

The name “Enable” refers to a level-sensitive input signal, whereas the name “Execute” refers o a edge-trigger input signal.

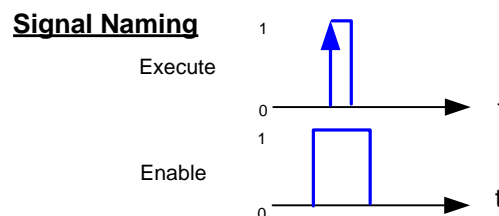


Figure 1: Signal Naming for edge- and level triggered input signals

1.2.5. Axis_REF Data type

The Axis_REF is a structure that contains information on the corresponding axis. It is used as a VAR_IN_OUT in all Function Blocks defined in this document. The content of this structure is implementation dependent and can ultimately be empty. If there are elements in this structure, the supplier shall support the access to it but this is outside the scope of this document. The refresh rate of this structure is also implementation dependent.

Axis_REF data type declaration:

```
TYPE AXIS_REF : STRUCT
    (Content is implementation dependent)
END_STRUCT
```

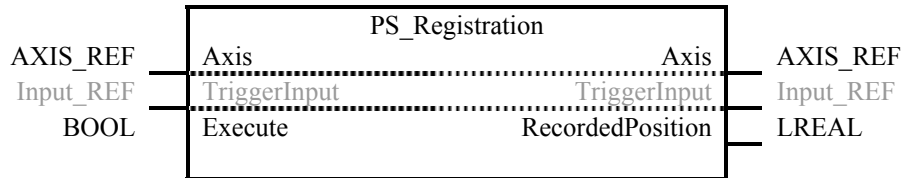
Example:

```
TYPE AXIS_REF : STRUCT
    AxisNo: UINT;
    AxisName: STRING(255);
END_STRUCT
```

1.2.6. Documentation Conventions

Variables and types of the implementation interface are listed in tables and a FB graphical representation. (The code implementation and the execution of the FB may be implemented in any IEC61131-3 language, see above). Optional (extended) Inputs and Outputs are represented in grey font. VAR_IN_OUT are represented dashed underlined inside the body of the function block as represented in this document.

Example:



1.2.7. Compliance and Portability

The objective of this work is to achieve a level of portability of Function Blocks acting on a base level device, and providing the same functionality to the user as described within this paper, with respect to user interface, input / output variables, parameters and units used.

The possibility of combining several MC libraries from different suppliers within one application is left open to be solved by the systems integrator.

An implementation which claims compliance with this specification shall offer a set of (meaning one or more) Function Blocks with at least the **basic** interface variables marked as “**B**” in the defined tables in the definition of the Function blocks in this document. For higher level systems and future extensions any subset of the **extended** interface variables marked as “**E**” in the tables can be implemented. **Supplier specific** additions should be marked “**S**”

Three sets of interface variables are defined:

1. Basic Set Shown in the tables with the letter “**B**”
2. Extended Set Shown in the tables with the letter “**E**”
3. Supplier Set Shown in the tables with the letter “**S**”

The basic set is focused to the minimal requirements to create a functional control system. The extended version defines the options for higher level systems, and future extensions. Compliance and usage of the OMAC PWG logos is described in Appendix A.

1.3. Certification

In order to fulfill the requirements set, different levels of certification are applicable:

1. Certification / Declaration of Conformity of the software tool supplier, often part of the control supplier
2. Certification / Conformity of the application at the user and/or machine builder

Ad 1: Certification /Conformity Declaration of the software tool supplier, often part of the control supplier

The development environment, including the Packaging related function blocks, need to be certified by the supplier according to rules set forth in IEC61131-3. These requirements are beyond the scope of this document. Typically, a supplier will self-certify the level of compliance of his product with the technical paper provided. A certification matrix is provided in the appendix to this document.

Ad 2: Certification / Conformity of the application at the user and/or machine builder

Within an application, a certification includes the Packaging related software combined with the infrastructure, like sensors, switches and actuators, connection schemes, etc. Certification or Conformity Declarations for these applications are beyond the scope of this document, and have to be dealt with by external involved entities.

The use of the OMAC or Packaging Workgroup logos does not give any guarantee about any compliance or fulfillment. The use of the logo just refers to the inclusion of the ideas and guidelines as described in this document, within the relevant software environment, and the availability of this information in more detail on the relevant section of OMAC website www.OMAC.org.

2. Overview of the defined Packaging Application Function Blocks

Packaging Process Functions	Machine Communication
Wind / Unwind Axis (constant surface velocity, csv mode)	PS_Send,
Wind / Unwind Axis (constant torque, ct mode)	PS_Receive
Dancer Control	PS_CommRequest
Registration	PS_Indicate
Registration_Correction	PS_CommRespond
Indexing	PS_CommConfirm
Batch Counter	PS_CommNotify
Digital PLS (Digital CAM Switch)	PS_CommIndicate
SetOverride	PS_Publish
JogAxis	PS_Subscribe
Flying Sync	
Gear_In with Dynamic Gear Factor	
Motion Command Stop with oriented halt	
Machine Behavior Organization	
PackML Machine State Model FB	

Table 1: Defined Function Blocks of PackAL

3. Packaging Application Function Blocks

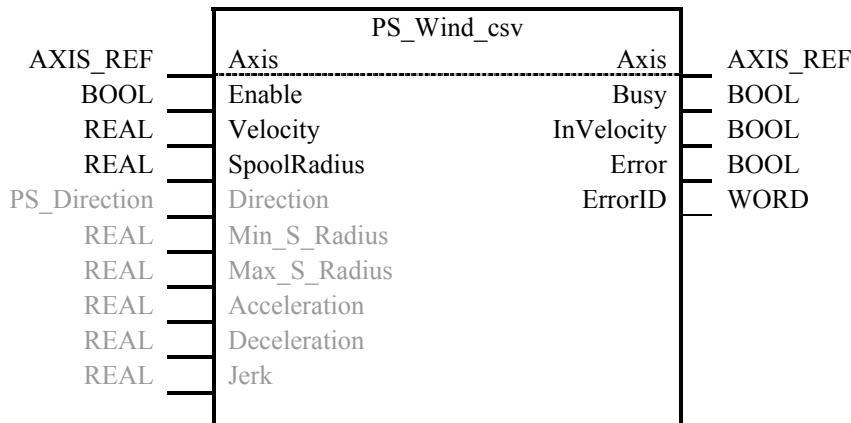
3.1. Wind / Unwind Axis (constant surface velocity, csv mode)

FB-Name		PS_Wind_csv	
This function block commands a controlled motion at a specified circumflex velocity for a wind / unwind axis. The circumflex is calculated from the radius of a wind/unwind spool, measured with a sensor.			
VAR IN OUT			
B	Axis	AXIS_REF	
VAR INPUT			
B	Enable	BOOL	Start the motion at high level, stop at low level
B	Velocity	REAL	Value of the maximum Surface velocity [u/s]
B	SpoolRadius	REAL	Actual value of the radius of the spool [u]
B	Min_S_Radius	REAL	Value of the Minimal Spool Radius, initial value [u]
B	Max_S_Radius	REAL	Value of the Maximal Spool Radius, initial value [u]
E	Direction	PS_Direction	Enum type (pos, neg)
E	Acceleration	REAL	Value of the acceleration (increasing energy of the motor) [u/s ²]
E	Deceleration	REAL	Value of the deceleration (decreasing energy of the motor) [u/s ²]
E	Jerk	REAL	Value of the Jerk [u/s ³]
VAR OUTPUT			
B	Busy	BOOL	Executing status
B	InVelocity	BOOL	command circumflex velocity phase active
B	Error	BOOL	Signals that error has occurred within Function block
B	ErrorID	WORD	Error Number

Note : This FB contains the mathematical calculation for coupling between a rotary slave axis and a translatory master axis. Its purpose is to generate and re-adjust a constant surface velocity (peripheral speed), relative to the master axis, for the rotary calibrated and controlled slave axis, depending on its spool diameter. Via a signal proportional to the spool radius, the spool radius of this slave axis is automatically evaluated and used for the calculation. This radius must never have the value 0.0 mm, since otherwise a calculation is no longer possible.

- The FB decelerates the axis to stop while winding above the Max_S_Radius
- The FB decelerates the axis to stop while unwinding below the Min_S_Radius
- SpoolRadius needs to satisfy Min_S_Radius < SpoolRadius < Max_S_Radius to execute the action, otherwise Error = True, ErrorID set
- Error ID is implementation specific

SpoolRadius is scanned in every cycle of the Function Block execution, other inputs in first cycle only.



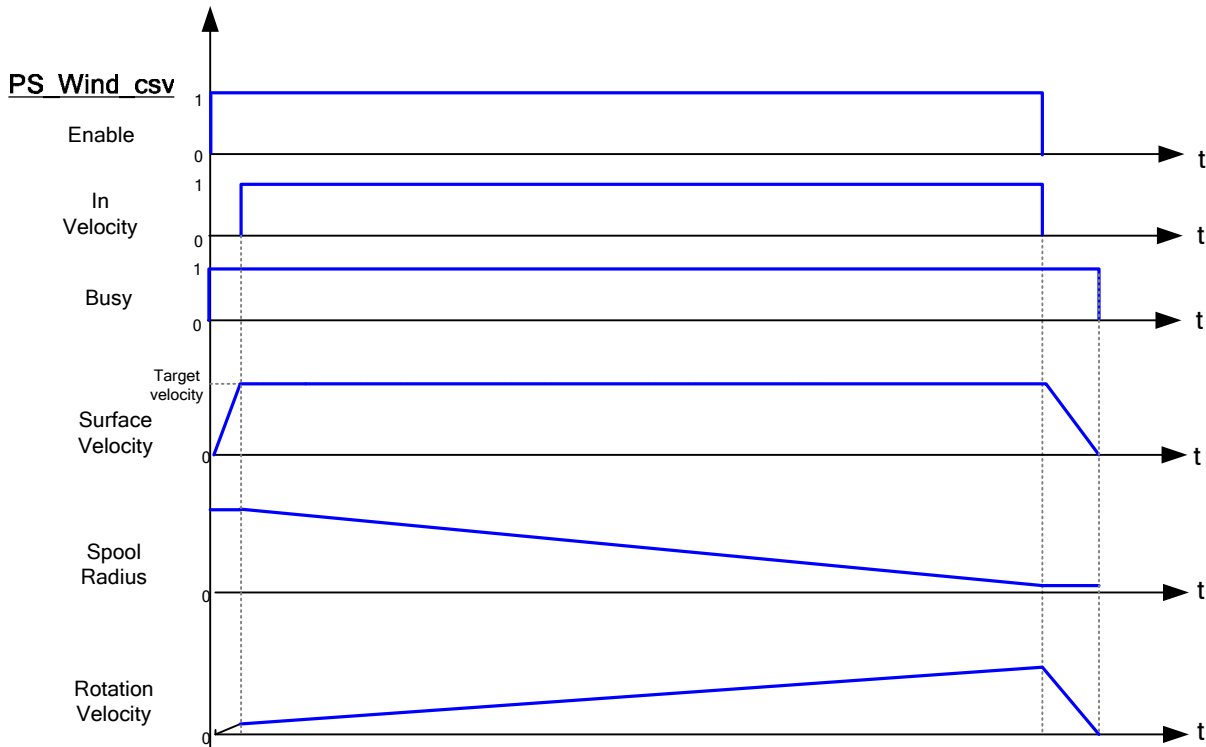


Figure 2: PS_wind_csv timing diagram

3.2. Wind / Unwind Axis (constant torque, ct mode)

FB-Name	PS_Wind_ct		
This function block commands a simpler controlled motion at a specified torque for a wind / unwind axis. The circumflex is calculated from the radius of a wind/unwind spool, measured with a sensor.			
VAR IN OUT			
B	Axis	AXIS_REF	Identifies the axis to work upon
VAR INPUT			
B	Enable	BOOL	Start the motion at high level, stop at low level
B	Torque	REAL	Value of the torque (Torque or force in t.u.)
B	SpoolRadius	REAL	Actual value of the radius of the spool [u]
B	Min_S_Radius	REAL	Value of the Minimal Spool Radius, initial value [u]
B	Max_S_Radius	REAL	Value of the Maximal Spool Radius, initial value [u]
E	TorqueRamp	REAL	The maximum time derivative of the torque (Torque or force in t.u. per sec)
E	Direction	PS_Direction	Enum type (pos, neg)
VAR OUTPUT			
B	Busy	BOOL	Executing status
B	InTorque	BOOL	Set value of torque reached
B	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	WORD	Error identification

Notes: : This FB contains the mathematical calculation for coupling between a rotary slave axis and a translatory master axis. Its purpose is to generate and re-adjust a constant surface velocity (peripheral speed), relative to the master axis, for the rotary calibrated and controlled slave axis, depending on its spool diameter. Via a signal proportional to the spool radius, the spool radius of this slave axis is automatically evaluated and used for the calculation. This radius must never have the value 0.0 mm, since otherwise a calculation is no longer possible. In this mode, the FB will act in a simpler way with constant torque based solutions to the wind / unwind problem, e.g. for a FI based implementation.

- Max_S_Radius and Min:S_Radius are assumed to be initial values for first calculation of gear between master and slave axis at start of FB.
- The FB decelerates the axis to stop while winding above the Max_S_Radius
- The FB decelerates the axis to stop while unwinding below the Min_S_Radius
- SpoolRadius needs to satisfy $\text{Min_S_Radius} < \text{SpoolRadius} < \text{Max_S_Radius}$ to execute the action, otherwise Error = True, ErrorID set
- Error ID is implementation specific

SpoolRadius is scanned in every cycle of the Function Block execution, other inputs in first cycle only.

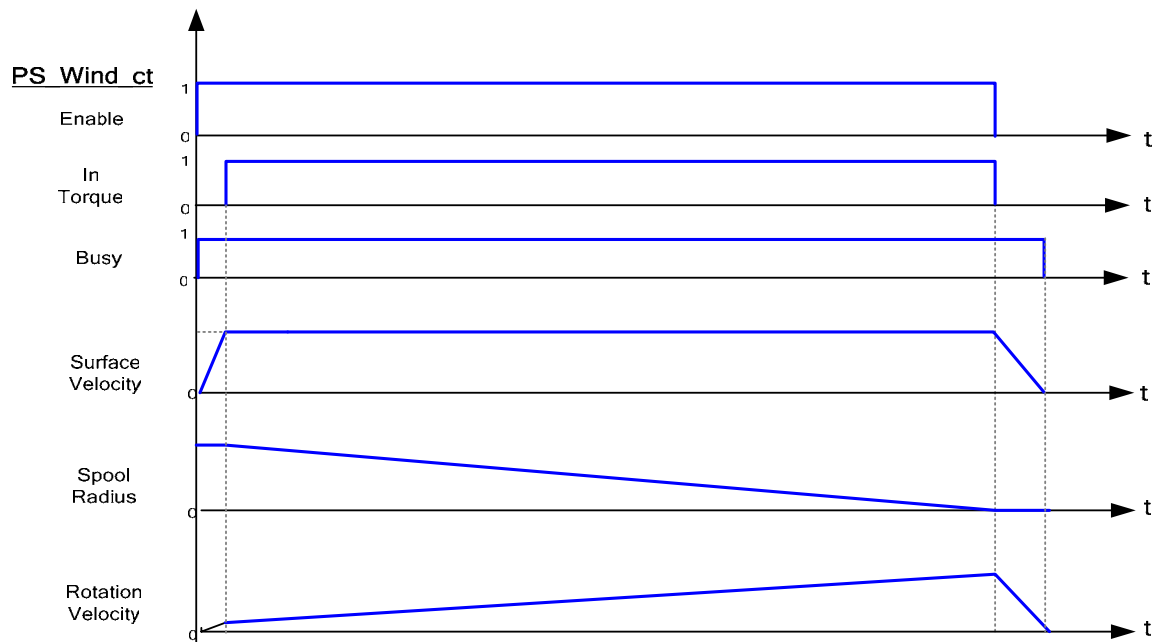
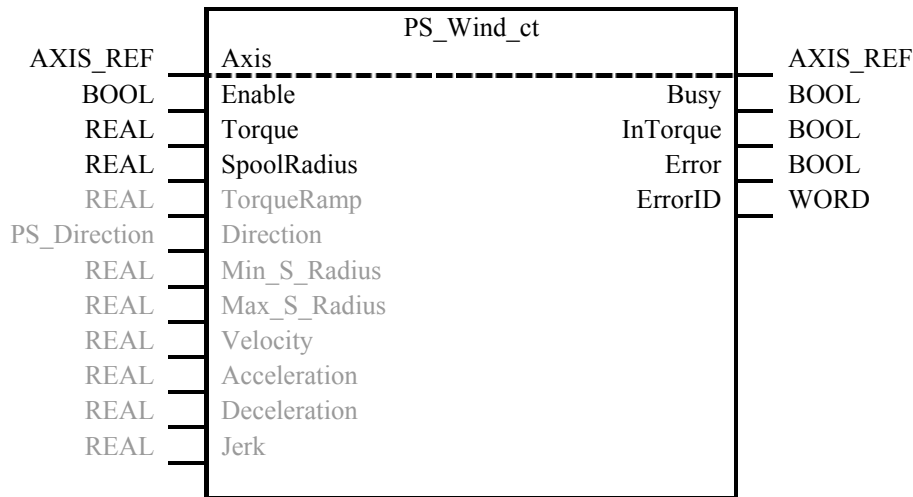


Figure 3: PS_Wind_ct timing diagram

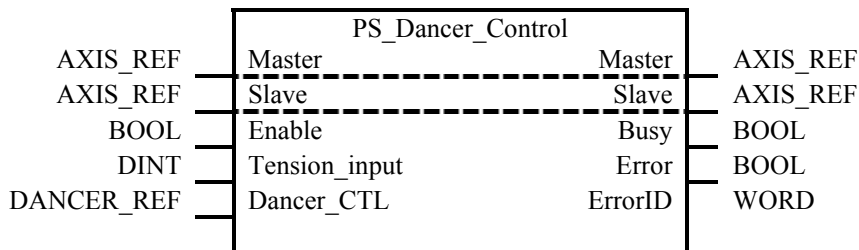
3.3. *Dancer Control*

FB-Name		PS_Dancer_Control		
This function block commands a controlled motion of an axis as slave of a dancer-coupled master axis. The Master axis is a physical or virtual axis. This FB provides the coupling between a slave axis (typically the infeed to the dancer) and a master axis (the outfeed of the dancer) via a dancer-PID controlled variable gearing factor.				
VAR_IN_OUT				
B	Master	AXIS_REF		
B	Slave	AXIS_REF		
VAR_INPUT				
B	Enable	BOOL	Start at high level, stop at low level	
B	Tension_input	REAL	Input signal for tension of the Web, usually represented by the relative actual dancer position sensor	
B	Dancer_CTL	DANCER_REF	Structure with Dancer Controller Parameters	
VAR_OUTPUT				
B	Busy	BOOL	Executing status	
B	Error	BOOL	Signals that an error has occurred within Function Block	
E	ErrorID	WORD	Error identification	

Notes : The FB’s purpose is to generate and re-adjust a constant surface velocity (peripheral speed), relative to the master axis, for the rotary calibrated and controlled slave axis, depending on a dancer position. Via a position signal (dancer position signal), the tension between master (web) and slave (e.g. spool) is represented. For general use, the raw dancer position is often aligned to a “balance position” by an offset (and optional multiplier) in a first dancer scaling algorithm. The PID calculates the control value depending on the difference to a scaled command dancer position. This PID control value output then tunes the gear ratio between the master (web) and the slave (e.g. spool). A multiplier limits the distortion of the gear, offset adjusts to gear setpoint. Scaling algorithm, dancer balance position, PID factors, gear factor, delta and offset to the gear are application specific.

Other possible implementations for Dancer Control, especially those with simpler two-switch control compared to PID control, may be represented by additional Function Blocks defined in the future.

Tension_input is scanned in every cycle of the Function Block execution, other inputs in first cycle only.



Elements within the array structure of Structure **Dancer_CTL**: DANCER_REF:

B/E	Parameter	Type	Description
B	Tension_ctl	DINT	Target value for web tension, typically the target position for the dancer in balanced condition
B	GearRatio	REAL	Ratio of gear factor g(t) between master (web) and Slave (spool) to balance the dancer. Default is 1.0.
B	deltaGear	REAL	Delta scaling multiplier of PID output (-1.0 ... +1.0) to GearRatio – must be smaller than 1 but never 0. deltagear would be e.g. 0.8 or 0.9 to limit the gear distortion

B	Gearoffset	REAL	scaling offset to fit GearRatio distortion by satisfying equation $g(t) = \text{deltaGear} * \text{PIDout} + \text{Gearoffset}$
B	fKp	REAL	PID Control Proportional gain (P)
B	fTn	REAL	PID Control Integral gain Tn (I) [s]
B	fTv	REAL	PID Control Derivative gain Tv (D-T1) [s]
B	fTd	REAL	PID Control Derivative damping time Td (D-T1) [s]
B	Accel_limit	REAL	corresponds indirectly, i.e. relative to a maximum master velocity, to a maximum permitted acceleration ($p_a = a_{\text{SlaveMax}} / v_{\text{MasterMax}}$). The limit parameter p_a corresponds to the reciprocal value of the run-up time $t_H = 1 / p_a$

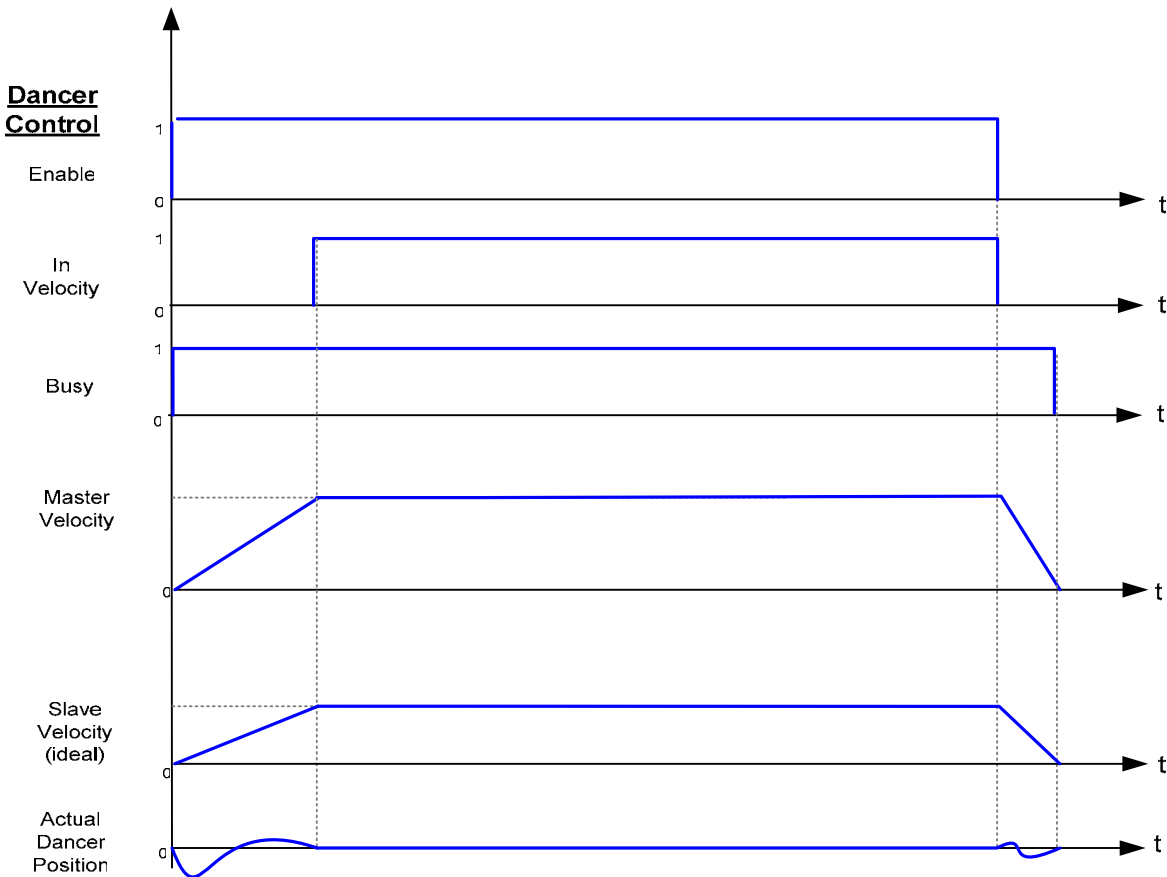


Figure 4: PS_Dancer Control timing diagram

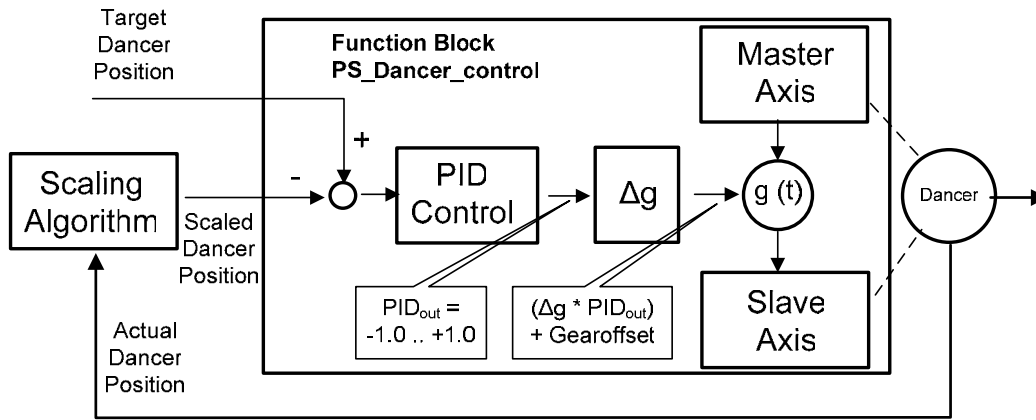


Figure 5: PS_Dancer Control Structure

3.4. Registration

FB-Name		PS_Registration	
This function block captures an axis position on receiving a rising edge execute command. The captured position is often referred to as “Latch Position”.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Identifies the axis which position shall be latched at the trigger event
E	TriggerInput	Input_REF	Reference to the trigger signal source. Trigger signal may be specified by the axis reference
VAR_INPUT			
B	Enable	BOOL	Start the Position Latch at high level
B	Recorded_Position	Position_REF	Actual Position of the Axis defined by Axis.
E	EnableWindow	BOOL	Enable an “capture input” window, in which input trigger events are captured
E	StartWindow	LREAL	Start position of capture window zone in technical units [u]
E	StopWindow	LREAL	Stop position of capture window zone in technical units [u]
VAR_OUTPUT			
B	Busy	BOOL	FB is active waiting for trigger event, capture pending on EnableWindow
B	RecordedPosition	LREAL	Latch Position, where the trigger event occurred (in t.u. [u])
B	PositionCaptured	BOOL	Trigger event recorded
B	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	WORD	Error identification
E	CommandAborted	BOOL	Command aborted
<p>Notes : This FB is a subset of the Registration_correction FB. Registration is often referred to as “Probing”. Some Motion Devices have dedicated “Registration” or “Probe” Inputs to quickly latch the position on the fly.</p> <p>Enable activates the Function Block, Recorded_Position is scanned in every cycle of the Function Block execution, other inputs in first cycle only.</p>			

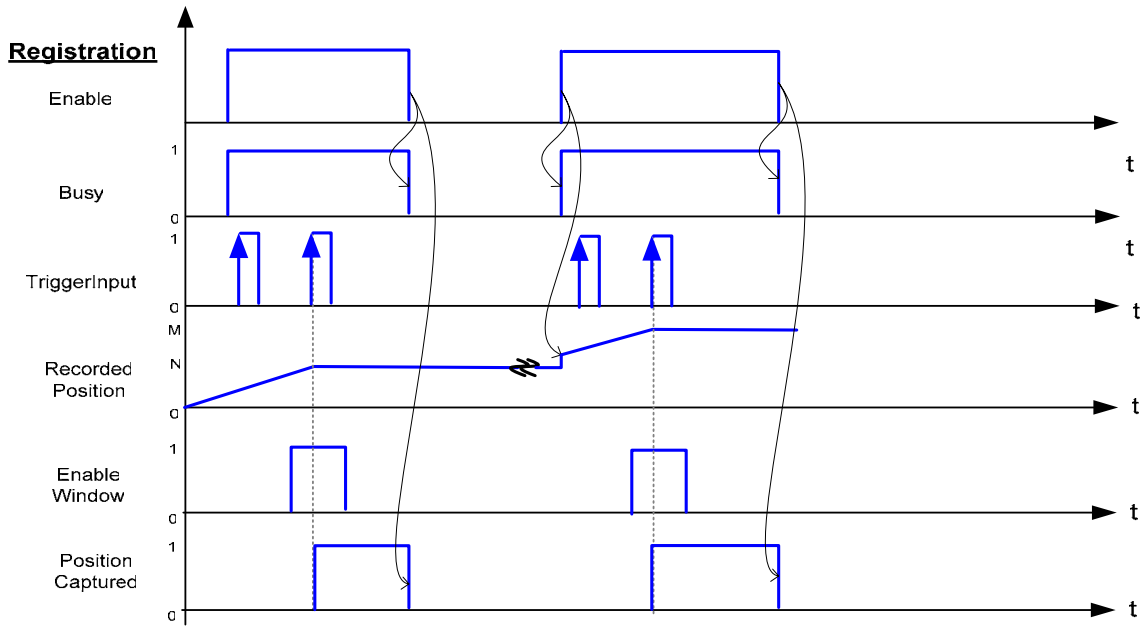
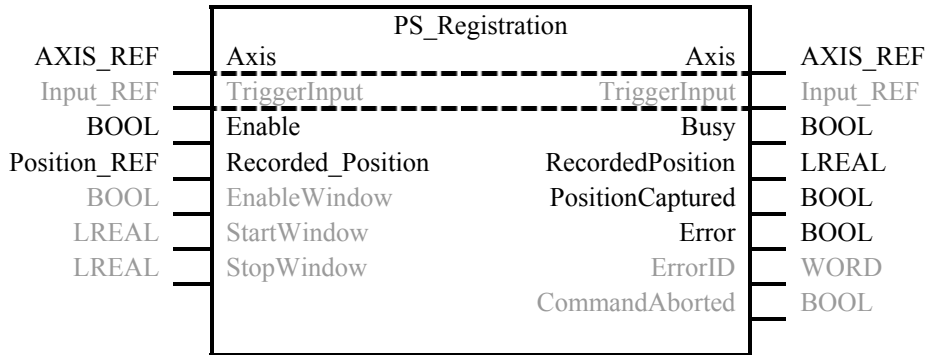


Figure 6: PS_Registration timing diagram

3.5. Registration_Correction

FB-Name		PS_Reg_Correction	
This function block corrects a distance offset, calculated after capturing an axis position, e.g. with the “Registration” FB, on receiving a rising edge execute command. The captured position is often referred to as “Latch Position”. The difference to the ideal position is the input “delta correction distance” to this FB.			
VAR IN OUT			
B	Master	AXIS_REF	
B	Slave	AXIS_REF	
VAR INPUT			
B	Enable	BOOL	Start the Correction FB, level triggered
E	Mode	BYTE	Correction method selector input. Two methods defined, other subject to supplier implementation
B	EnableCorr	BOOL	Level sensitive, TRUE during correction motion
B	DeltaCorr	LREAL	Correction distance (relative position) for correction move
B	CompZone	LREAL	Distance range of master in which correction move is to complete

E	DelayDistance	LREAL	Distance of master to move before correction starts
E	AccComp	LREAL	Max. acceleration for compensation [u/s ²]
E	DecComp	LREAL	Max. deceleration for compensation [u/s ²]
E	VdeltaComp	LREAL	Max. velocity for compensation move [u/s]
E	Vprocess	LREAL	Actual process (master) velocity [u/s]
E	TimeOut	TIME	Timeout supervision of the compensation move

VAR_OUTPUT

B	Busy	BOOL	This output remains TRUE as long as Function Block executes
B	Done	BOOL	This output changes from FALSE to TRUE once the correction motion is correctly done. It resets with the rising edge of EnableCorr or with Enable = FALSE.
B	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	WORD	Error identification

Notes : This correction FB executes after e.g. a registration FB captures apposition and the delta to a desired position has been determined by this or other methods. The mode selector allows the user to choose different ways of correction. Partial implementation of the solutions is possible; in this case, selection of the other methods results in an error.

Mode: 0 (Offset correction)

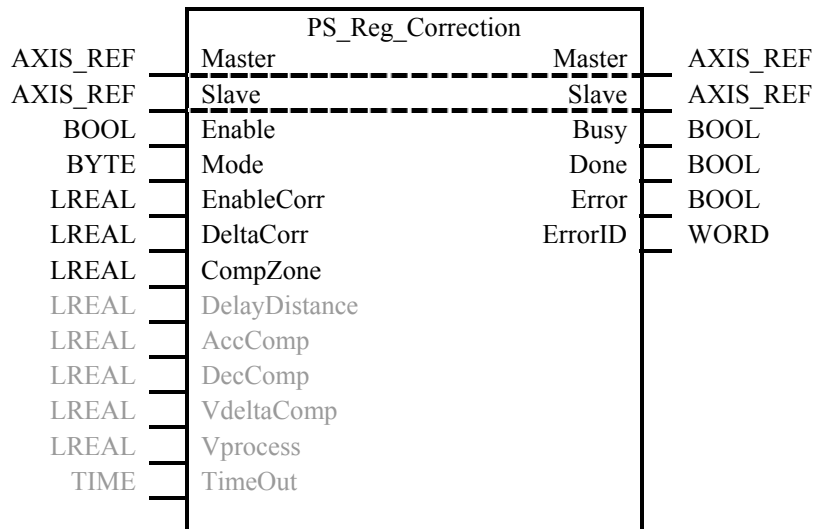
In this case, the DeltaCorr distance is directly applied to the Master Axis by means of a relative offset of the zero position, as used for CAM axes.

Mode: 1 (Distance correction)

The DeltaCorr distance is used to calculate a single super-positioned relative move of the slave axis to correct the delta correction distance.

Other correction methods may apply, e.g. by dynamic gear factor, and could be implemented as supplier specific enhancements.

Vprocess, the actual master of process velocity, is scanned in every cycle of the execution of the Function Block. Other inputs are scanned once at start of FB.



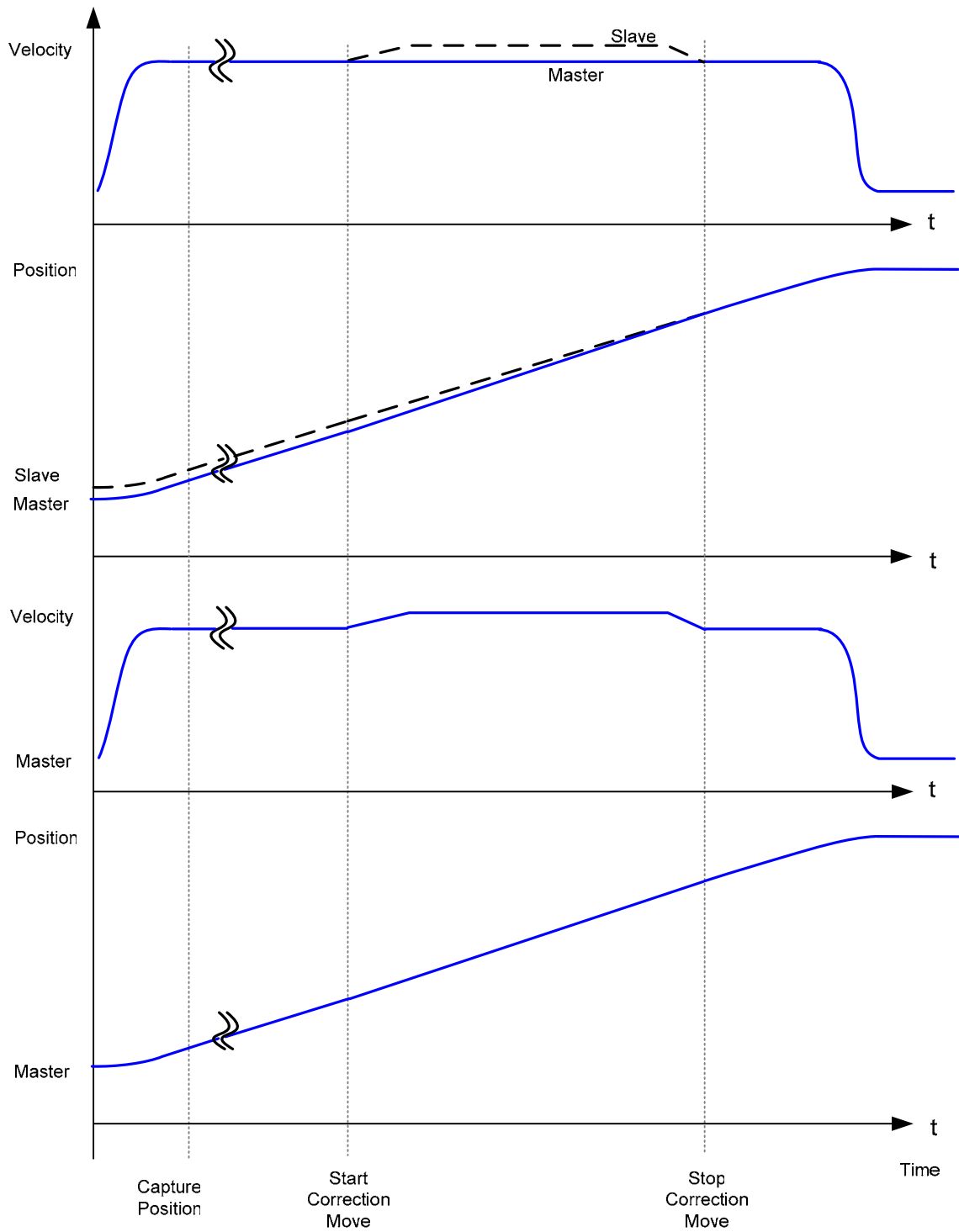


Fig. 3

Figure 7: PS_Reg_Correction timing diagram

Top: Distance correction mode
 Bottom: Offset correction mode
 blue: Master, black: Slave

3.6. Indexing

FB-Name		PS_Indexing	
This function block will – upon R_Trigger on Execute – do a number of relative moves, listed in a table (Struct or Array). The FB will position the axis to a complete stop at target position and continue with the next move from the table upon next R_Trigger signal.			
VAR IN OUT			
B	Axis	AXIS_REF	
VAR INPUT			
B	Execute	BOOL	Start the next relative positioning by a rising edge.
B	Positions	POS_REF	Reference to Structure or Array with relative move distances listed. Typical Type of Positions: LREAL
B	Mode	INT	Mode of Indexing, see notes
E	Table_pointer	INT	Pointer to element n of Table
VAR OUTPUT			
B	Index_No	INT	Index executing or last index completed
B	Busy	BOOL	This output remains TRUE until the block has executed a command
B	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	WORD	Error identification

Notes : This function block will – upon R_Trigger on Execute – do a number of relative moves 1 - N, listed in a table (Struct or Array). The FB will position the axis to a complete stop at target position n and continue with the next move from the table upon next Execute signal.

Modes of operation are

- Mode: 0 pass through table elements once until finding zero distance entry,
- Mode: 1 cycle through table, starting at first element on finding a zero distance entry,
- Mode: 2 position relative for distance of table element N (Table_pointer)

The FB will complete a singular move if possible. In case a second Execute rising edge triggers the next move, the FB will not decelerate to stop but continue to new target position immediately.

Inputs are updated on rising edge of Execute.

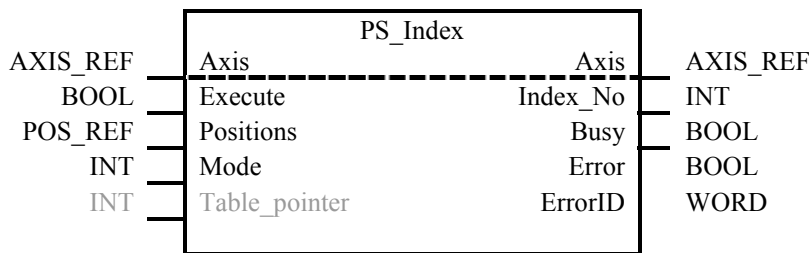


Table **Positions:**

Position 1	Velocity 1	Acceleration 1	Deceleration 1	Jerk 1
Position 2	Velocity 2	Acceleration 2	Deceleration 2	Jerk 2
...
Position N	Velocity N	Acceleration N	Deceleration N	Jerk N

Table 2: PS_Index Position table

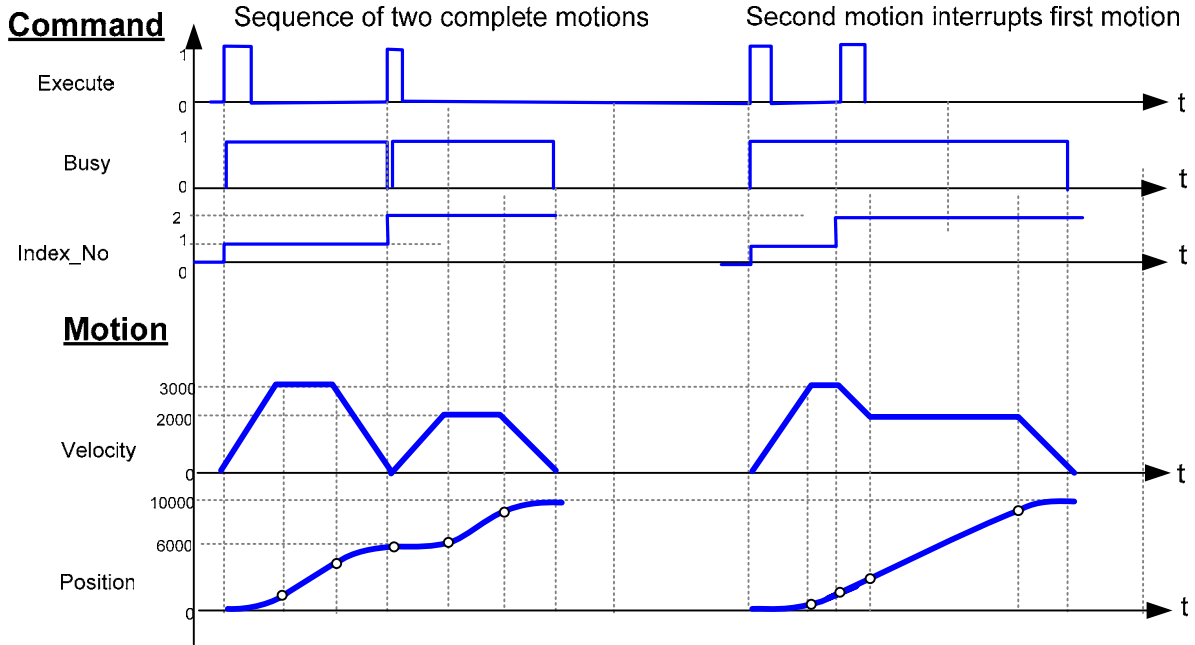
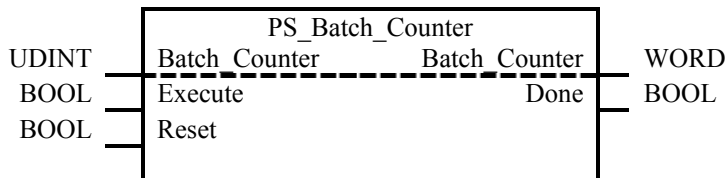


Figure 8: PS_Indexing timing diagram

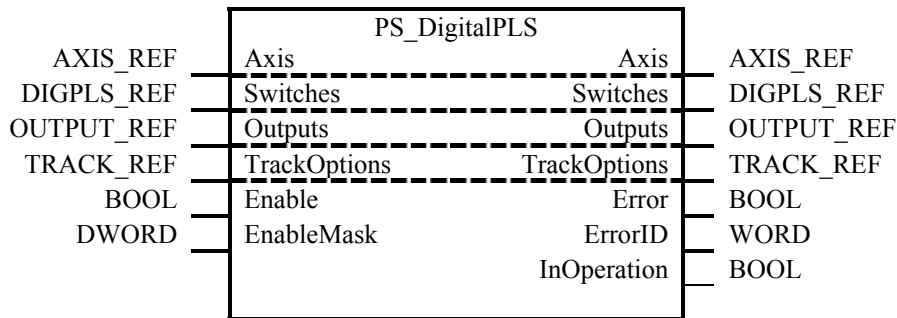
3.7. Batch Counter

FB-Name		PS_Batch_Counter	
This function block provides a simple batch counter to handle the number of process cycles to accomplish a batch.			
VAR_IN_OUT			
B	Batch_counter	UDINT	Batch counter, decrementing
VAR_INPUT			
B	Execute	BOOL	Decrements the batch counter to zero on R_trig
B	Reset	BOOL	Resets batch counter to zero
VAR_OUTPUT			
B	Done	BOOL	True if batch counter is zero
Notes : This function block will – upon rising edge of Execute – decrement the Batch_counter by 1 to zero and set the “Done” flag. A reset input allows to reset the counter to 0 for an early termination. Batch_Counter is set to the number of cycles to go and the FB counts down on every cycle. Once the Batch Counter reaches 0, the DONE Flag is set and the decrement stops on further Execute commands.			



3.8. Digital PLS (Phase Limit Switch or CAM Switch)

FB-Name		PS_DigitalPLS	
This function block is the analogy to switches on a motor shaft: it commands a group of discrete output bits to switch in analogy to a set of mechanical cam controlled switches connected to an axis. Forward and backward movements are allowed. Note that it is intended to keep this FB compatible with PLCopen MC FB Extensions (Part 2) MC_CAMSWITCH function Block.			
VAR IN_OUT			
B	Axis	AXIS_REF	Reference to the axis to which the switches are connected to
B	Switches	DIGPLS_REF	Reference to the switching actions.
B	Outputs	OUTPUT_REF	Reference to the signal outputs, directly related to the referenced tracks. (max. 32 per function block) (First output = first TrackNumber)
E	TrackOptions	TRACK_REF	Reference to structure containing track related properties, e.g. the ON and OFF compensations per output/track.
VAR INPUT			
B	Enable	BOOL	Enables the Switches outputs
E	EnableMask	DWORD	32 bits of BOOL. Enables the different tracks. Least significant data is related to the lowest TrackNumber. With data SET (to '1' resp. TRUE) the related TrackNumber is enabled.
VAR OUTPUT			
B	Error	BOOL	Signals that error has occurred within Function block
E	ErrorID	WORD	Error Identification
B	InOperation	BOOL	The commanded tracks are enabled
Notes:			
<ul style="list-style-type: none"> • DIGPLS_REF is a supplier specific reference to the pattern data. • OUTPUT_REF is a supplier specific structure linked to the (physical) outputs • TRACK_REF is supplier specific structure containing the track properties, e.g. the compensation per track (A track is a set of switches related to one output). 			



Basic elements within the structure of **DIGPLS_REF**

B/E	Parameter	Type	Description
B	TrackNumber	INT	TrackNumber is the reference to the track
B	FirstOnPosition [u]	REAL	Lower boundary where the switch is ON
B	LastOnPosition [u]	REAL	Upper boundary where the switch is ON
E	AxisDirection	INT	Both (=0; Default); Positive (1); Negative (2)
E	CamSwitchMode	INT	Position based (=0; Default); Time based (=1)
E	Duration	TIME	Coupled to time based CamSwitchMode

Table 3: Digital PLS Structure table

Basic elements within the array structure of TRACK_REF

B/E	Parameter	Type	Description
E	OnCompensation	TIME	Compensation time with which the switching on is advanced or delayed in time per track.
E	OffCompensation	TIME	Time compensation the switching off is delayed per track.
E	Hysteresis [u]	REAL	Switching is shifted to a later reached position in order to avoid multiple switching around the switching point.

Table 4: Digital PLS Track table

This definition of cams has a start and an end position. So the user can define each single cam, that has an **FirstOnPosition** and an **LastOnPosition** (or time). In addition to a mechanical cam it has the possibilities to set it for a certain time and to give it a time compensation and a hysteresis. If (FirstOnPosition > LastOnPosition) it gives an inverse cam switch, which is off only within these positions.

CamSwitchMode can be Position, Time or other additional supplier specific types.

Duration: Time, the output of a time cam is ON

The time compensation (**OnCompensation** and **OffCompensation**) can be positive or negative. Negative means the output changes before the switching position is reached.

Hysteresis: This parameter avoids that the output is switching very often, if the axis is near the switching point and the actual position is jittering around the switching position. Hysteresis is part of TRACK_REF, which means that for each track a different hysteresis is possible.

Example of CAMSWITCHREF

Parameter	Type	Switch01	Switch02	Switch03	Switch04	...	SwitchN
TrackNumber	INTEGER	<i>1</i>	<i>1</i>	<i>1</i>	<i>2</i>		
FirstOnPosition [u]	REAL	<i>2000</i>	<i>2500</i>	<i>4000</i>	<i>3000</i>		
LastOnPosition [u]	REAL	<i>3000</i>	<i>3000</i>	<i>1000</i>	<i>--</i>		
AxisDirection	INTEGER	<i>1=Pos</i>	<i>2=Neg</i>	<i>0=Both</i>	<i>0=Both</i>		
CamSwitchMode	INTEGER	<i>0=Position</i>	<i>0=Position</i>	<i>0=Position</i>	<i>1=Time</i>		
Duration	TIME	<i>--</i>	<i>--</i>	<i>--</i>	<i>1350</i>		

Note: Values are Examples

Table 5: Digital PLS Switch Position table

Example.

This example uses the values from the example for DIGPLS_REF above. It uses no On/OffCompensation, nor hysteresis. This is the behavior of the outputs, when the axis is moving continuously in positive direction. The axis is a modulo axis with a modulo length of 5000 u.

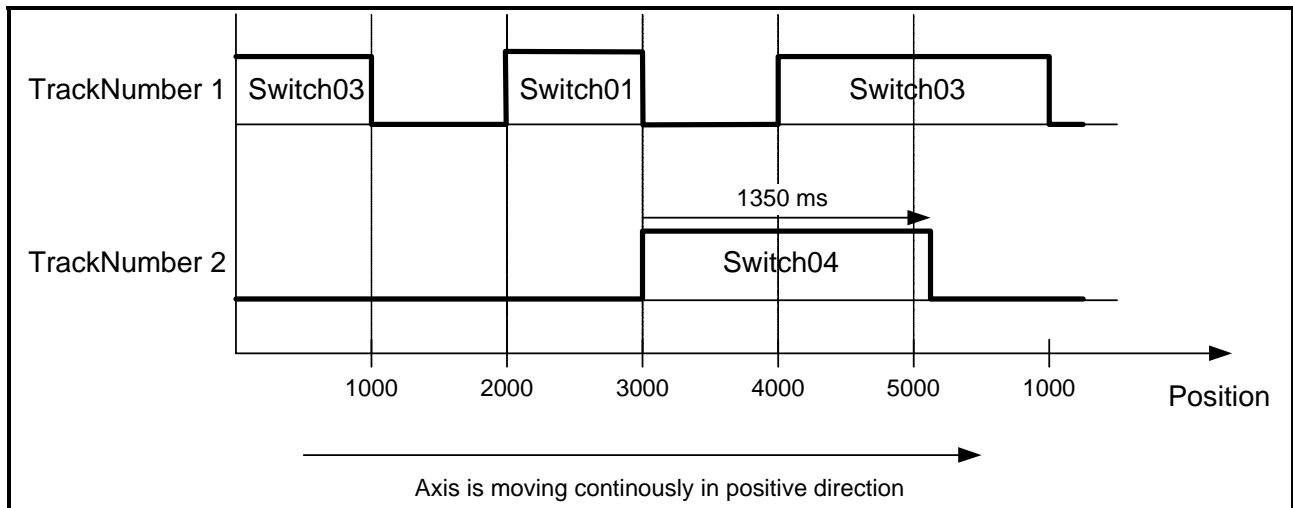


Figure 9: Digital PLS timing diagram

Detailed description of Switch01.

This example additionally uses OnCompensation -125ms and OffCompensation +250ms.

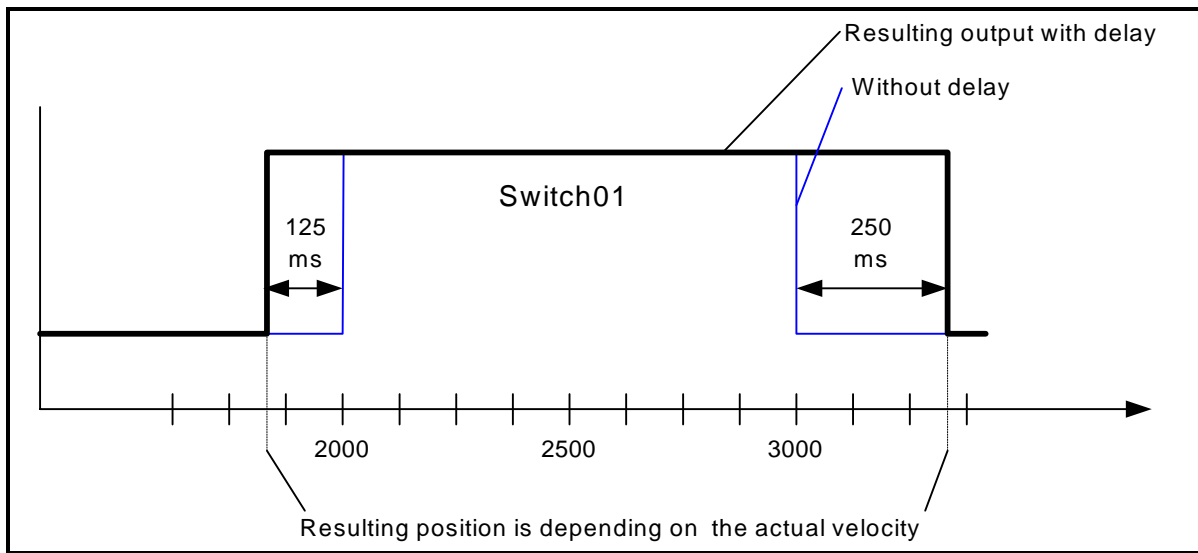


Figure 10: Digital PLS, detailed description of Switch01.

This is the behavior of the outputs, when the axis is moving continuously in negative direction without On- or OffCompensation and without Hysteresis.

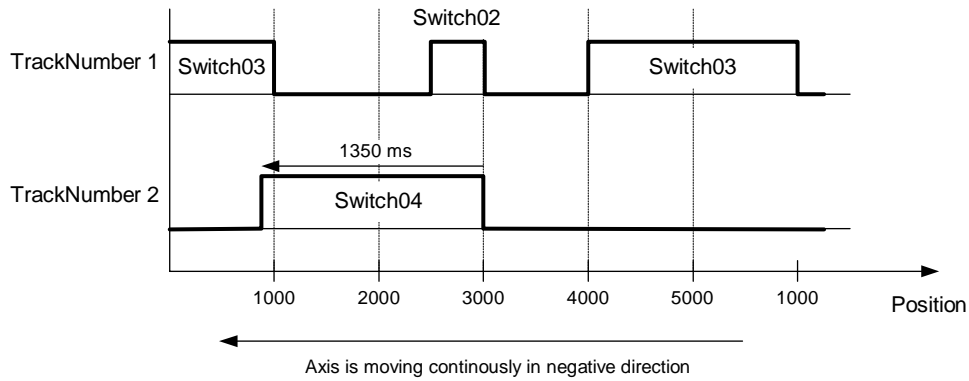
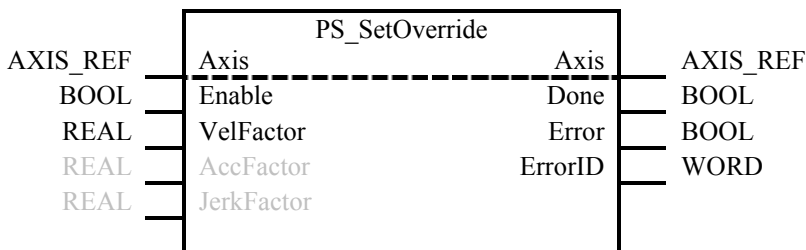


Figure 11: Digital PLS duration switch timing diagram

3.9. Override

FB-Name		PS_SetOverride	
This function block sets the values of override. The override parameters act as a factor that is multiplied to the commanded velocity, acceleration, deceleration and jerk of the move function block. Note that it is intended to keep this FB compatible with PLCopen MC FB Extensions (Part 2) MC_SetOverride function Block.			
VAR_IN_OUT			
B	Axis	AXIS_REF	Identifies the axis to work upon
VAR_INPUT			
B	Enable	BOOL	Write the value of the override factor at rising edge
B	VelFactor	REAL	New override factor for the velocity (e.g. 0...100 %)
E	AccFactor	REAL	New override factor for the acceleration/deceleration
E	JerkFactor	REAL	New override factor for the jerk
VAR_OUTPUT			
B	Done	BOOL	Signals that the override factor(s) is (are) set successfully
B	Error	BOOL	Signals that error has occurred within Function block
E	ErrorID	WORD	Error identification
Notes:			
<ol style="list-style-type: none"> The Input AccFactor acts on positive and negative acceleration (deceleration). This FB sets the factor. The override factor is valid until a new override is set. The default values of the override factor are 1.0. The value of the overrides can be between 0.0 and 1.0. The behavior of values > 1.0 is supplier specific. Values < 0.0 are not allowed. The value 0.0 is not allowed for AccFactor and JerkFactor. The value 0.0 set to the VelFactor stops the axis without bringing it to the state standstill. Override does not act on slave axes. (Axes in the state synchronized motion). The FB does not influence the state diagram of the axis. Override can be changed at any time and acts directly on the ongoing motion. 9. Override overrides all other motion commands active in regards to velocity applied 			



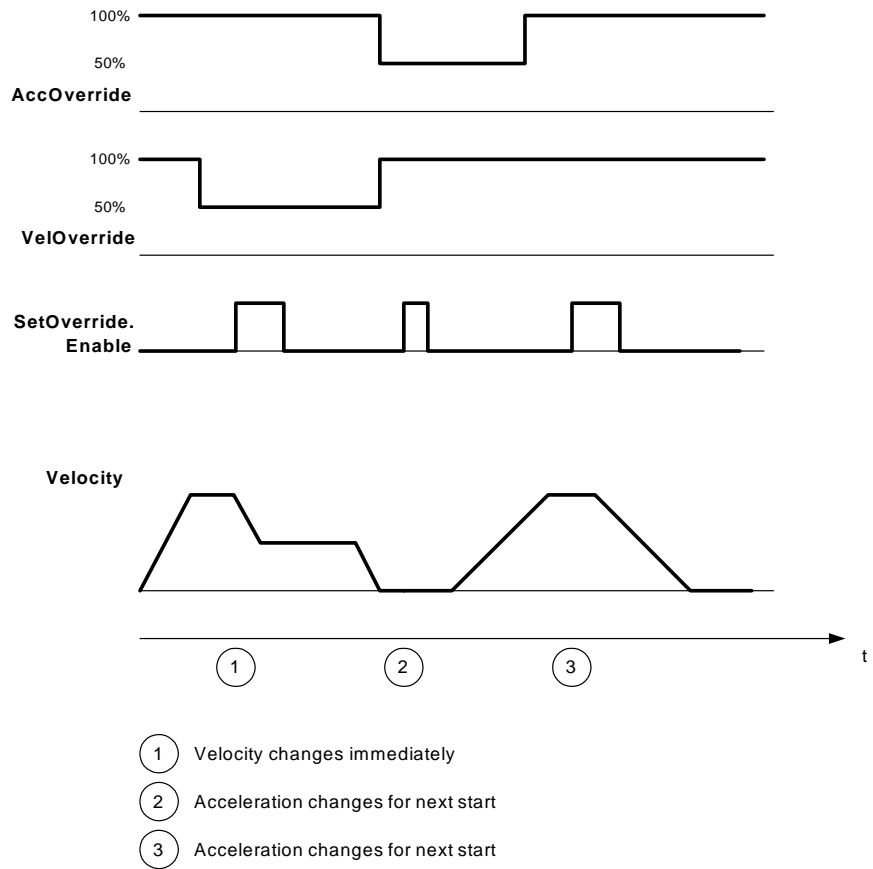


Figure 12: PS_SetOverride timing diagram

3.10. JogAxis

FB-Name		PS_JogAxis	
<ul style="list-style-type: none"> This function block jogs an axis for N increments (represented in t.u.) forward or backward with the selected (manual operation) jog velocity and acceleration. N must satisfy certain minimum count limitation, otherwise no movement is executed. 			
VAR IN OUT			
B	Axis	AXIS_RE F	Reference to axis to be jogged
VAR INPUT			
B	Enable_Forward	BOOL	Executes a sequence of a relative move Forward during high and a stop at signal = low
B	Enable_Backward	BOOL	Executes a sequence of a relative move Backward during high and a stop at signal = low
B	JogVelocity	REAL	Velocity to jog (in t.u. [u])
B	Min_Jog_Distance	REAL	Minimum distance to jog (in t.u. [u])
VAR OUTPUT			
B	Busy	BOOL	True if jogging is being performed
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new jog command while still executing
E	ErrorID	WORD	Error identification

Notes : This function block will – after rising edge on Enable_Forward or Enable_Backward - start a continuous move (at least for the minimum distance) and continue upon high-level on these inputs with a continuous motion – until they are FALSE – then, on their falling edge, the axis is regularly decelerated to stop. The movement is carried out on Jog velocity for the minimum distance or longer.

- In case both Enable signals are high, the FB will assume the result to be low as in an EXOR conjunction.

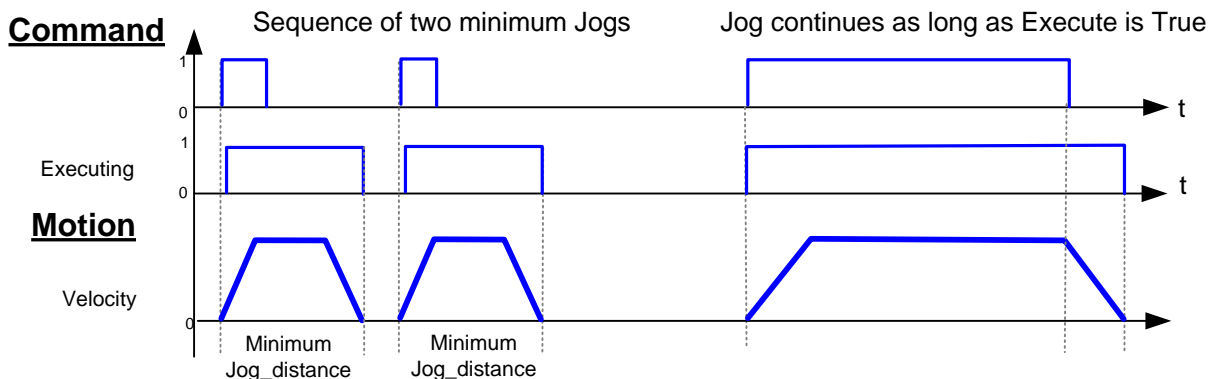
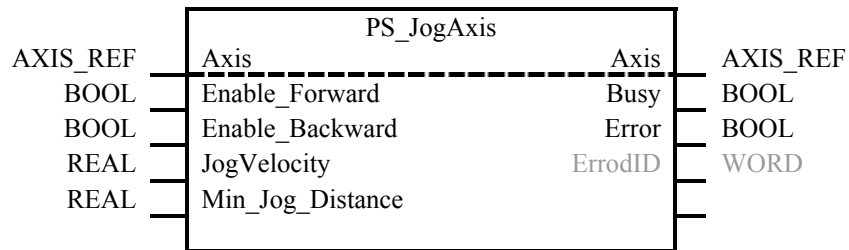


Figure 13: PS_Jog Axis timing diagram

3.11. Flying Sync

The Flying Sync function provides a slave axis that can be synchronized to a moving master axis comparable to a “flying shears” or alternatively “flying cutter” functionality. The slave axis moves in synchronism with the master axis to perform machining processes. This kind of movement, synchronized to the master axis, allows to machine a work piece even while it is being transported.

The "Flying Sync" function is able to start synchronization of the slave even when the slave has already started, and is therefore no longer stationary. The "Flying Sync" function also calculates improved set value profiles, and these can be influenced by the user through a wide range of boundary conditions.

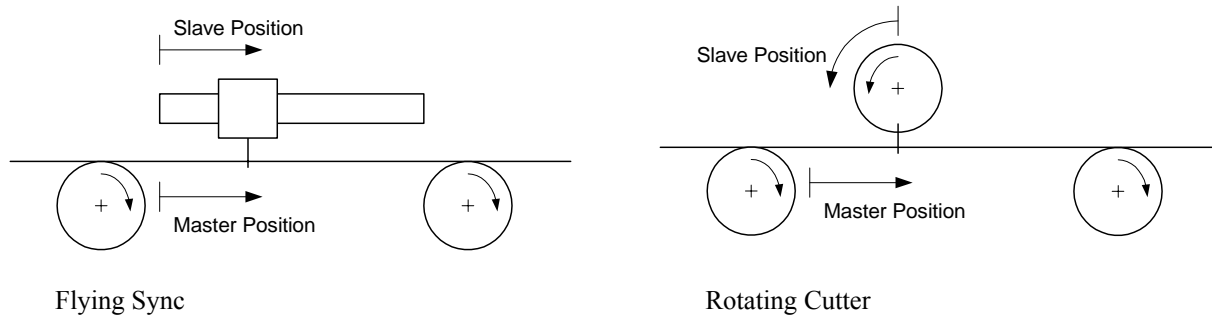


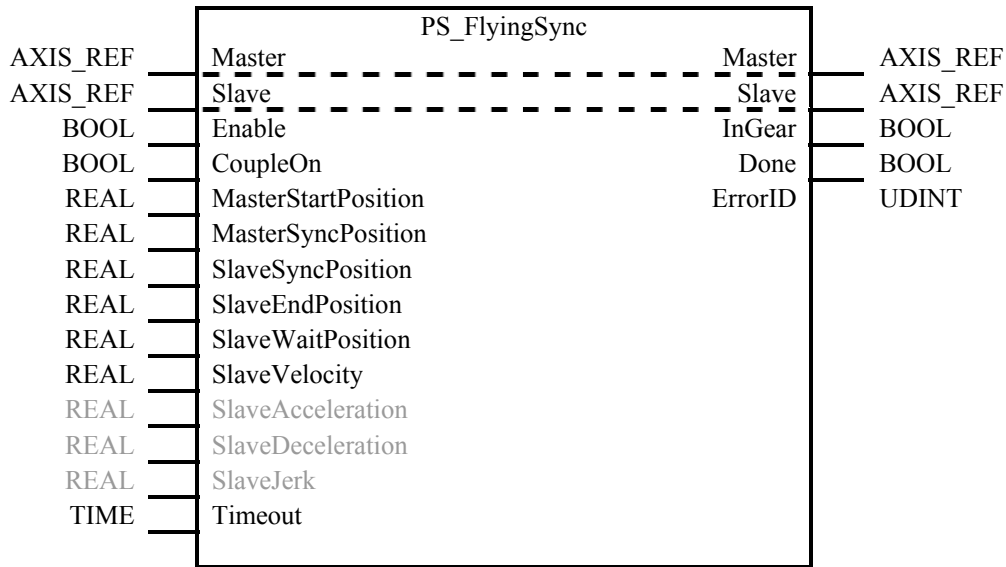
Figure 14: Flying Sync Geometries

FB-Name		PS_FlyingSync	
<ul style="list-style-type: none"> PS_FlyingSync couples a slave axis to a master axis similar to a flying shear or flying saw. The synchronization speed is achieved precisely at the synchronization position and kept for the distance to the SyncEndposition. The FB allows to either provide a 			
VAR IN OUT			
B	Axis_Slave	AXIS_REF	Reference to axis to be coupled as a flying saw
B	Axis_Master	AXIS_REF	Reference to axis to be coupled to as Master for a flying saw
VAR INPUT			
B	Enable	BOOL	Initiates the function block and moves slave initially to SlaveWaitPosition
B	CoupleOn	BOOL	Couple initiation, level sensitive -> loads MasterSyncPosition and SlaveSyncPosition for execution. On FALSE, coupling is disabled
B	MasterStartPosition	REAL	Master position that determines the phase relation between master and slave axis [u]
B	MasterSyncPosition	REAL	Master position where synchronized motion starts [u]
B	SlaveSyncPosition	REAL	Corresponding slave position [u]
B	SlaveEndPosition	REAL	Slave position where synchronized part of motion ends [u]
B	SlaveWaitPosition	REAL	Slave position where slave axis waits [u]
B	SlaveVelocity	LReal	Value of the maximum slave velocity (always positive) (not necessarily reached) [u/s].
E	SlaveAcceleration	LReal	Value of the acceleration (always positive) (increasing energy of the motor) [u/s ²]
E	SlaveDeceleration	LReal	Value of the deceleration (always positive) (decreasing energy of the motor) [u/s ²]
E	SlaveJerk	LReal	Value of the Jerk [u/s ³]. (always positive)
B	Timeout	TIME	Supervisory timeout

VAR_OUTPUT			
B	InSync	BOOL	True if slave is in sync with Master
B	Done	BOOL	True once Slave is in wait position
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new command while still executing
B	ErrorID	UDINT	Error identification

Notes :

- Slave retracts after Slave reaches SyncEnd position to Wait Position
- Slave might either work in finite travel space (for flying shears function) by retracting to wait position
- Slave might work in infinite travel space (for cutter function) by traveling forward only
- If Slave Accel, Decel and Jerk are not supported, default values will apply.



The figures above give a sketch that shows the function to be solved with this function block. The primary function is to cut a passing continuous flow of material at specified positions into discrete pieces (i.e. products). Optionally, the FB allows to apply a Slave Wait Position larger than Slave End Position to create the function of a rotating cutter.

Both functions are quite similar, with the difference that for the Flying Shear the slave axis features a finite range of motion and for the rotating cutter the slave axis continues to move in one direction.

The motion can be split into two parts:

1. Synchronization between master and slave axis and synchronous motion of both axes (where the actual task is performed)
2. Slave axis moves to a defined waiting position and waits for next action request

For the first part, the master and slave axes have to operate at synchronous speeds while maintaining a specified phase relationship (e.g. to assure an accurate cutting point). With this function block, the sequence of actions can be easily generated. Figure 2 shows a timing diagram of this process.

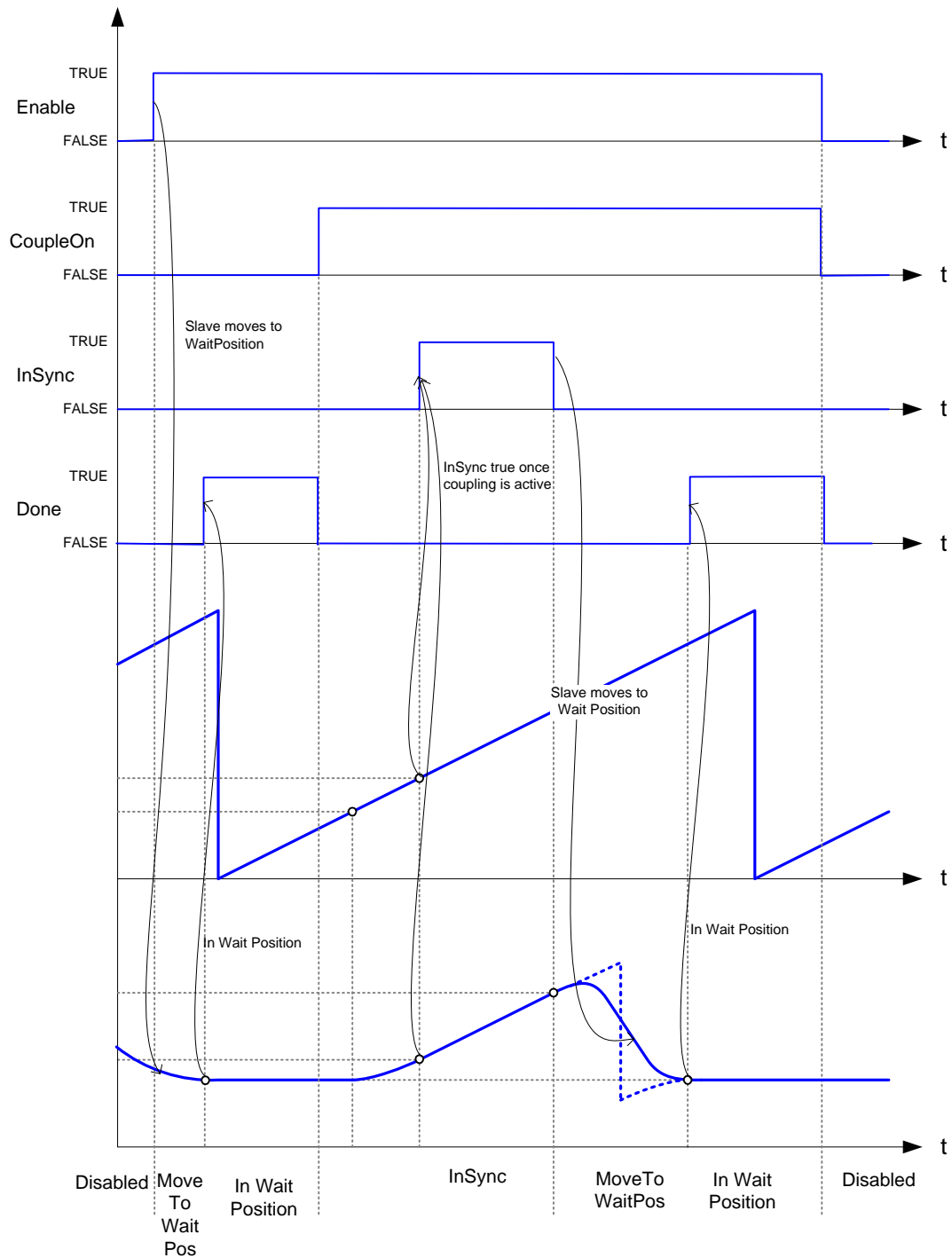


Figure 15: PS_FlyingSync timing diagram for a single synchronization

3.12. Gear_In with dynamic Gear Factor

FB-Name		PS_GearInDyn	
<ul style="list-style-type: none"> PS_GearInDyn activates a linear master-slave coupling (gear coupling). The gear ratio can be modified in every control cycle. The block is therefore suitable for velocity controlled master slave couplings. The acceleration parameter limits the acceleration of the slave in case of high gear ratio changes. 			
VAR_IN_OUT			
B	Axis Master	AXIS_REF	Reference to axis to be coupled to as Master
B	Axis Slave	AXIS_REF	Reference to axis to be coupled
VAR_INPUT			
B	Enable	BOOL	Coupling initiation, level sensitive. Initiates coupling and maintains during high, aborts on low signal
B	GearRatio	LREAL	Gearing factor. The gear ratio may be changed in every PLC cycle. 0.0 is invalid, default is 1.0
B	Acceleration	LREAL	Limit to acceleration of slave due to gear ration changes
B	Deceleration	LREAL	Limit to deceleration of slave due to gear ration changes
VAR_OUTPUT			
B	Busy	BOOL	Becomes TRUE, as soon as the block is active. Busy becomes TRUE with a rising edge at Enable and becomes FALSE again after the block finished or aborted its operation. A high level at Enable will then be accepted.
B	InGear	BOOL	True if coupling is being successfully performed
B	CommandAborted	BOOL	True if slave is decoupled during Enable is TRUE
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new command while still executing
E	ErrorID	UDINT	Error identification
Notes :			

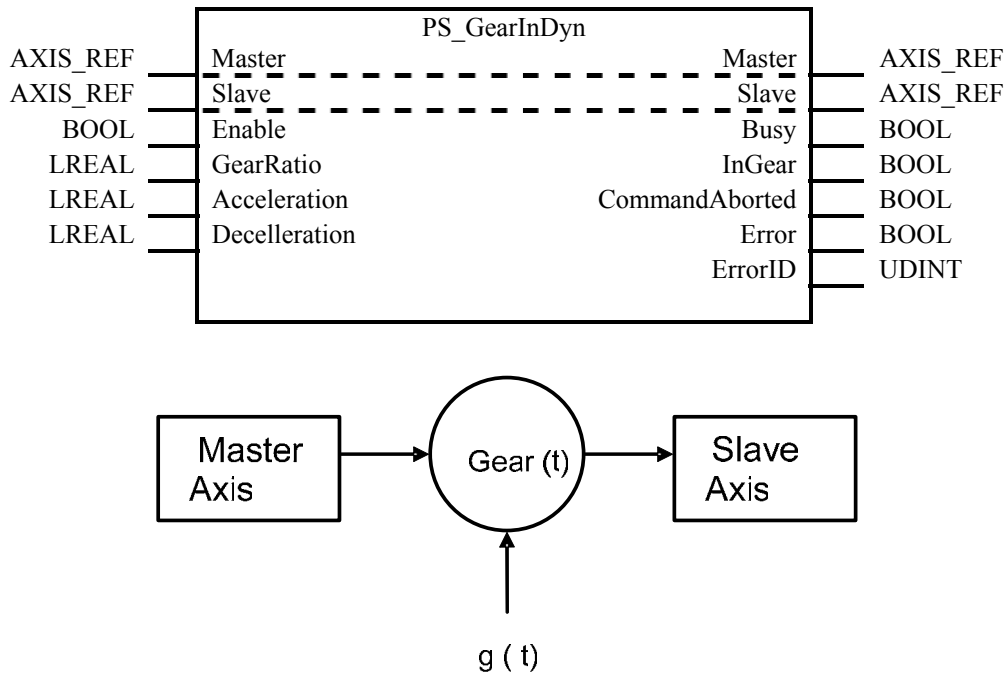
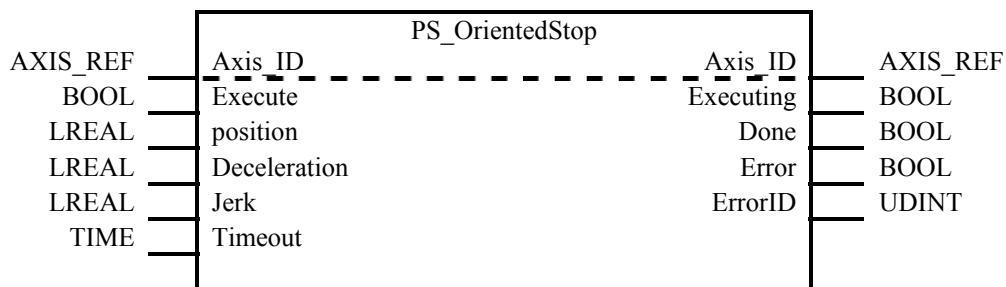


Figure 16: PS_GearInDyn structure of gear model

3.13. Motion Command Stop with oriented Halt

FB-Name		PS_OrientedStop	
<ul style="list-style-type: none"> PS_OrientedStop is used to affect an "oriented" axis stop at the specified modulo position. Depending on the speed and the dynamic parameters of the axis, it will stop at the next possible oriented position. <p>For convenience in packaging, a stop with a resulting orientation of tool or product is often utilized. Therefore, this stop command calculates the deceleration ramp to final stop but will end always in the desired modulo position.</p>			
VAR_IN_OUT			
B	Axis_ID	AXIS_REF	Reference to axis to be stopped
VAR_INPUT			
B	Execute	BOOL	Stop initiation, rising edge trigger
B	fposition	LREAL	Absolute modulo destination position at which the axis should stop
E	fDeceleration	LREAL	Limit to deceleration of axis during decel ramp. This input is optional. The default deceleration specified at the start of the axis is used for stopping.
B	fJerk	LREAL	Limit to jerk of of axis during decel ramp. This input is optional. The default jerk specified at the start of the axis is used for stopping.
B	Timeout	TIME	Supervisory timeout
VAR_OUTPUT			
B	Executing	BOOL	True if coupling FB is being executed
B	Done	BOOL	True if stop was successfully performed and the axis has come to a complete stop
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new command while still executing
E	ErrorID	UDINT	Error identification
<p>Notes : The error output must be analyzed, because under certain conditions an oriented stop is not possible. For example, a standard stop may have been triggered just before, or an oriented stop would cause an active software limit switch to be exceeded. For all fault conditions, the axis is stopped safely, but it may subsequently not be at the required oriented position.</p> <p>The position must be in the range $[0 \leq \text{position} < \text{modulo period}]$. With a modulo period of 360 degrees, for example, 360 is therefore not a valid position. A value of 0 has to be specified instead.</p>			



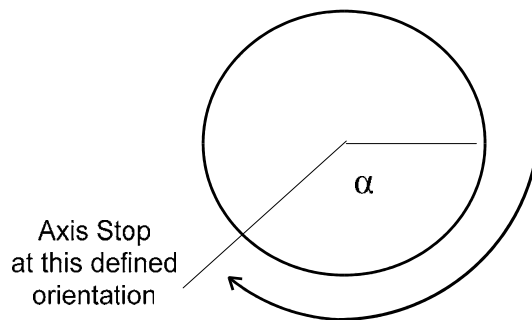


Figure 17: PS_OrientedStop graphical explanation of behavior

4. Packaging Machine State Blocks

Packaging Machine State Function Blocks provide a common interface to the existing PackML Machine State Model implementations available. It is expected that the application specific logic including the transitions between states is programmed in external function blocks, but the central logic of the state machine and the status representation should be handled by the Pack_ML_State_Machine Function block. Therefore, this FB comes with a recommendation how to combine with other logic.

4.1. Generic Machine State Function Block

4.2. PS_PackML_State_Machine

FB-Name		PS_PackML_State_Machine	
<ul style="list-style-type: none"> PS_PackML_State_Machine executes the PackML State Machine. Transitions need be programmed depending on machine application; however, the machine status and sequence of transitions shall satisfy the State Model listed for reference. 			
VAR_IN_OUT			
E	Machine_ID	Machine_REF	Reference to machine State Model runs on
VAR_INPUT			
B	Execute	BOOL	Execute State Machine from rising edge
B	Abort	BOOL	to Aborting or to Aborted
B	Stop	BOOL	to Stopping or to Stopped
B	Start	BOOL	to Starting
B	Ready	BOOL	to Ready
B	Standby	BOOL	to Standby
B	Produce	BOOL	to Producing
B	Hold	BOOL	to Holding or to Held
B	Off	BOOL	to Off
E	Auto_Mode	BOOL	True if automatic mode is selected, otherwise FALSE
VAR_OUTPUT			
B	Status	WORD	Status Word representing the status of the State Machine
B	ST_Aborting	BOOL	True if State Machine is in state Aborting
B	ST_Stopping	BOOL	True if State Machine is in state Stopping
B	ST_Stopped	BOOL	True if State Machine is in state Stopped
B	ST_Off	BOOL	True if State Machine is in state Off
B	ST_Starting	BOOL	True if State Machine is in state Starting
B	ST_Ready	BOOL	True if State Machine is in state Ready
B	ST_Standby	BOOL	True if State Machine is in state Standby
B	ST_Producing	BOOL	True if State Machine is in state Producing
B	STj_Holding	BOOL	True if State Machine is in state Holding
B	ST_Held	BOOL	True if State Machine is in state Held
E	ST_Manual	BOOL	True if Manual mode is selected
E	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	UDINT	Error identification

Notes :

For enhancement of the use for this state model, some variables were added as Extension Types:

To identify the machine executed by the state model, a Machine ID is implemented for convenient reference. Also, to switch from automatic mode to manual mode, a selector input is given. Error indications will be displayed by an Error ID. All inputs are scanned continuously except for Machine_ID, which is latched at rising edge of Execute.

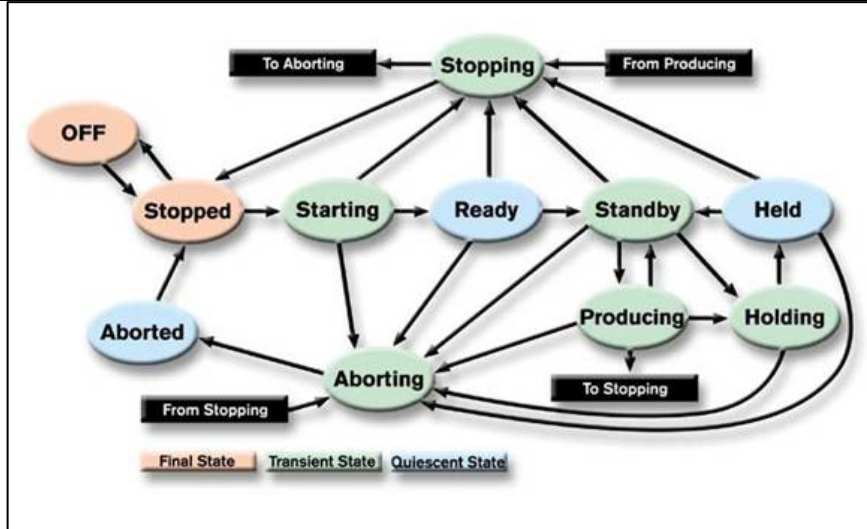


Figure 18: PackML State Model

PS_PackML_State_Machine			
Machine_REF	Machine_ID	Machine_ID	Machine_REF
BOOL	Execute	Status	WORD
BOOL	Abort	ST_Aborting	BOOL
BOOL	Stop	ST_Stopping	BOOL
BOOL	Start	ST_Stopped	BOOL
BOOL	Ready	ST_Off	BOOL
BOOL	Standby	ST_Starting	BOOL
BOOL	Produce	ST_Ready	BOOL
BOOL	Hold	ST_Standby	BOOL
BOOL	Off	ST_Producing	BOOL
BOOL	Auto_Mode	ST_Holding	BOOL
		ST_Held	BOOL
		ST_Manual	BOOL
		Error	BOOL
		ErrorID	UDINT

The state transitions to a machine application are always application specific. Therefore, to facilitate standardization, it should be best practice to form state function blocks that are connected to the PS_PackML_State_Machine. The state function blocks collect application specific signals and form the transition logic to the adjacent states as displayed in the PackML state model. All state FB feed back into PS_PackML_State_Machine, offering a standard state machine and state reporting. The state FBs will contain the machine execution code next to the application specific transition logic.

State Function Blocks are listed in Figure N and will be programmed by application programmer accordingly to maintain integrity and function of the PackML State Machine:

PS_Pack__ML_State_Model Function Block Names:

PS_AbortingState	PS_StandbyState
PS_StoppingState	PS_ProducingState
PS_StoppedState	PS_HoldingState
PS_OffState	PS_HeldState
PS_StartingState	PS_ManualState
PS_ReadyState	

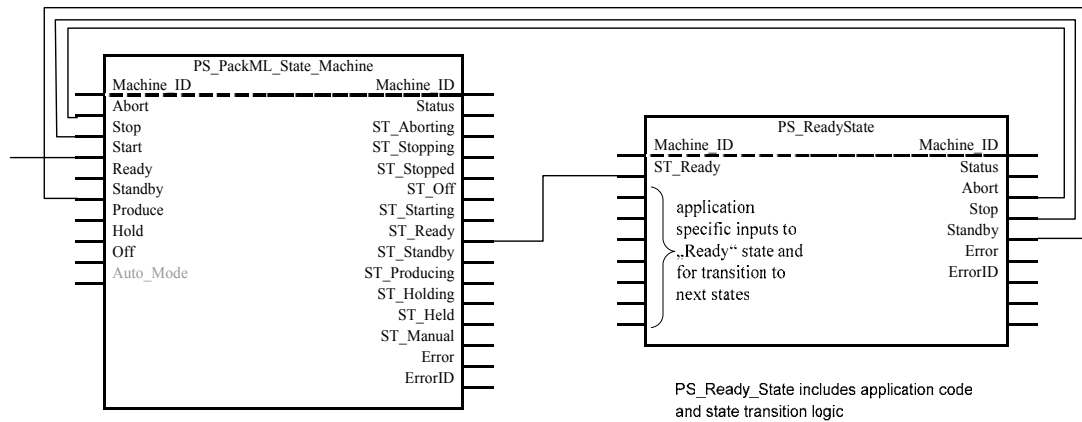


Figure 19: PS_Pack_ML_State_Model FB Names and application adoption

PackML_State_Machine and application specific state Function Blocks (e.g., but not limited to, PS_ReadyState) containing state logic and machine execution code. All other states require according Function Blocks to contain the application adoption. Therefore, above listed Function Block names are reserved for this state model:

5. Packaging Communication Function Blocks

Packaging communication Function blocks provide basic methods to communicate between cell controllers. Three types of functions are defined:

- Send/ Receive Data FB for basic (serial) Point-to-Point communications,
- A confirmed handshake for Telegram based communications, as known with many protocols, to cover multi-Point communication,
- Publish /Subscribe for variable publishing based communications as alternative multi-point communication.

Intention is to standardize these three basic communications to be used independently of network standards to abstract from certain specific network technologies to allow software to run regardless of underlying network offered.

5.1. Point to Point Communication by Function Block

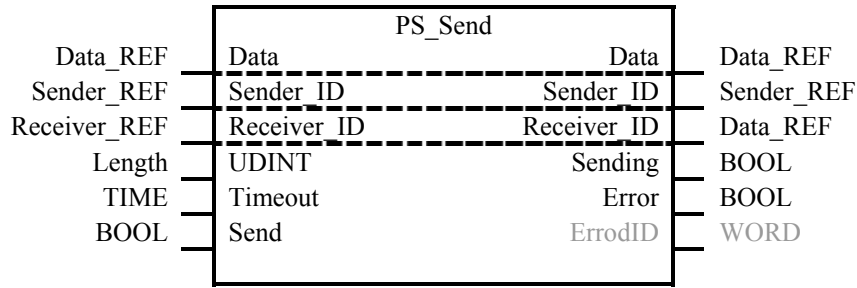
Provides references to data elements e.g. variables or structures to communicate the content of these data elements. PackTags can be handed from control to control or to devices by means of these higher level function blocks.

PS_Send, PS_Receive

Provide simple methods of communication over cell bus or other media. Sender and Receiver ID must be known, the communication is a point to point communication.

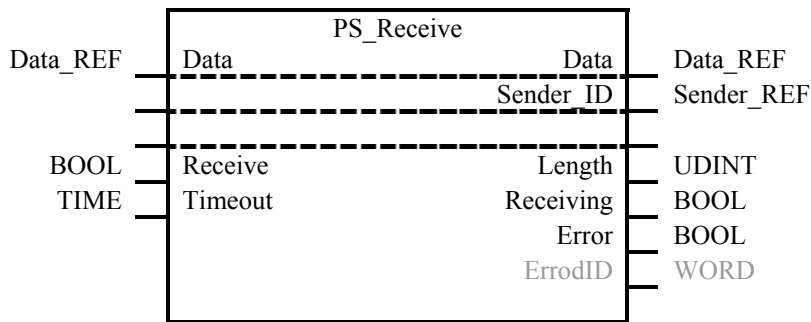
5.1.1. PS_Send

FB-Name		PS_Send	
<ul style="list-style-type: none"> • This function block sends the data structure referenced to the receiver. This may take more than one cycle of control or network of communication. 			
VAR_IN_OUT			
B	Data_REF	Data_REF	Reference to data to be sent
B	Sender_ID	Sender_REF	Reference to Sender_ID
B	Receiver_REF	Receiver_RE F	Reference to Receiver_ID
VAR_INPUT			
B	Length	UDINT	Length of data set in bytes
B	Send	BOOL	Executes a send sequence
B	Timeout	TIME	Supervisory timeout
VAR_OUTPUT			
B	Busy	BOOL	True if sending is being performed
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new send command while still executing
E	ErrorID	WORD	Error identification
Notes :			



5.1.2. PS_Receive

FB-Name		PS_Receive	
<ul style="list-style-type: none"> This function block receives data structures referenced to the receiver. This may take more than one cycle of control or network of communication. The FB is executed in every control cycle 			
VAR IN_OUT			
B	Data_REF	Data_REF	Reference to data to be stored after reception
B	Sender_ID	Sender_REF	Reference to Sender_ID retrieved from transmission
VAR INPUT			
B	Receive	BOOL	Awaiting reception of data
B	Timeout	TIME	Supervisory timeout
VAR OUTPUT			
B	Length	UDINT	Length of data set in bytes retrieved from Message
B	Receiving	BOOL	True if receiving data
B	Error	BOOL	Signals that an error has occurred within Function Block
E	ErrorID	WORD	Error identification
Notes :			



5.2. Message or Telegram based Communication

Establishes a confirmed connection between devices. The communication is message or telegram based, transferring data content in a transparent way. All information, e.g. sender and receiver header information, is encapsulated in the data field of implemented telegram in a transparent way. These FBs will not handle protocol specific content but manage raw telegram exchange, thus tunneling any specific information between sender and receiver.

The progress of this communication begins with an CommRequest, which arrives at the server where it invokes a CommIndication. The server replies with a CommResponse, which in turn is received by the client and creates a CommConfirmation flag.

Client	Telegram Transmission	Server
Request ->	-> Request telegram	-> Indication
Confirmation <-	<- Response telegram	<- Response

The sender creates a Communication Request, triggering the initiation of the communication by a first telegram. The Server receives the telegram, which creates an indication, and calculates a reply, transmitted by the response FB. The receipt of the answer telegram creates a confirmation at the client side. The communication is carried out by two telegrams in a confirmed way.

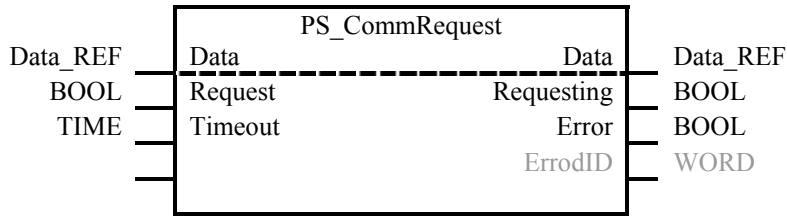
Messages which are sent autonomously by a server (e.g. error or other status messages) are registered by the client as a notification indication:

Client	Telegram Transmission	Server
Notification Indication <-	<- Notification telegram	<- Notification

The sender creates a Notification, which arrives at the Client without a previous Request handle. Therefore, it is noted as a Notification, forming an unconfirmed communication of data.

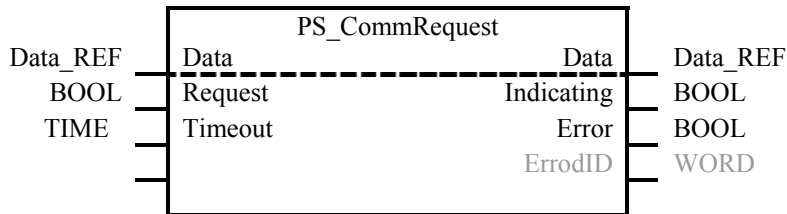
5.2.1. PS_CommRequest

FB-Name		PS_CommRequest		
<ul style="list-style-type: none"> This function block is used by a client and initiates a confirmed connection via a PS_CommunicationRequest The FB is executed once and triggers the request telegram. 				
VAR_IN_OUT				
B	Data_ID	Data_REF	Reference to Data_ID	
VAR_INPUT				
B	Request	BOOL	Executes a Request sequence to establish a connectin	
B	Timeout	TIME	Supervisory timeout	
VAR_OUTPUT				
B	Requesting	BOOL	True after FB Executed; cleared by Termination or Loss of connection (Timeout)	
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing	
E	ErrorID	WORD	Error identification	
Notes :				



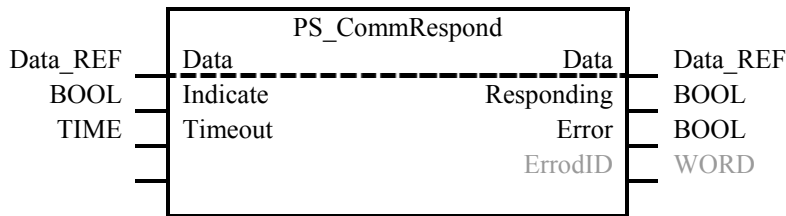
5.2.2. PS_Indicate

FB-Name		PS_CommIndicate	
<ul style="list-style-type: none"> This function block is executed by a server and responds to a CommRequest by a client, indicating availability to communicate to establish a confirmed connection. The FB is executed once. 			
VAR_IN_OUT			
B	Data_ID	Data_REF	Reference to Data_ID
VAR_INPUT			
B	Request	BOOL	Executes a Request sequence to establish a connectin
B	Timeout	TIME	Supervisory timeout
VAR_OUTPUT			
B	Indicate	BOOL	True after FB Executed; cleared by Termination or Loss of connection (Timeout)
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing
E	ErrorID	WORD	Error identification
Notes :			



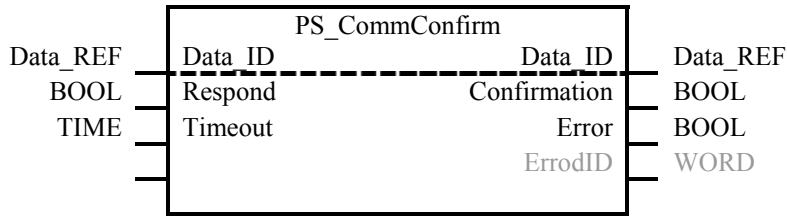
5.2.3. PS_CommRespond

FB-Name		PS_CommRespond		
<ul style="list-style-type: none"> This function block is executed by a server and responds back to a CommIndication by transmission of a telegram, confirming the establishing of a confirmed connection by a response. The FB is executed once and triggers the response telegram. 				
VAR IN_OUT				
B	Data_ID	Data_REF	Reference to Data_ID	
VAR INPUT				
B	Indicate	BOOL	Triggers a Respond sequence to establish a connection, edge-trigger	
B	Timeout	TIME	Supervisory timeout	
VAR OUTPUT				
B	Respond	BOOL	True after FB Executed; cleared by Termination or Loss of connection (Timeout)	
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing	
E	ErrorID	WORD	Error identification	
Notes :				



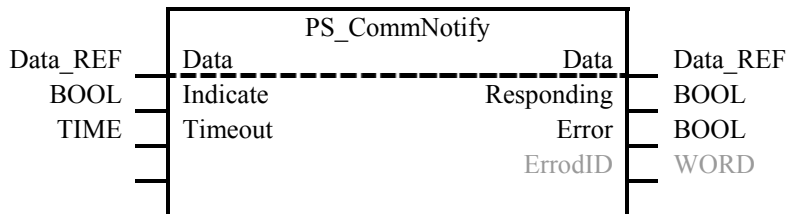
5.2.4. PS_CommConfirm

FB-Name		PS_CommConfirm		
<ul style="list-style-type: none"> This function block is executed by the client and terminates the communication from a CommResponse, handling the received data. The FB is executed once. 				
VAR IN_OUT				
B	Data_ID	Data_REF	Reference to Data_ID	
VAR INPUT				
B	Respond	BOOL	Triggers a Confirmation sequence to establish a connection, edge-trigger	
B	Timeout	TIME	Supervisory timeout	
VAR OUTPUT				
B	Confirmation	BOOL	True after FB Executed; cleared by Termination or Loss of connection (Timeout)	
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing	
E	ErrorID	WORD	Error identification	
Notes :				



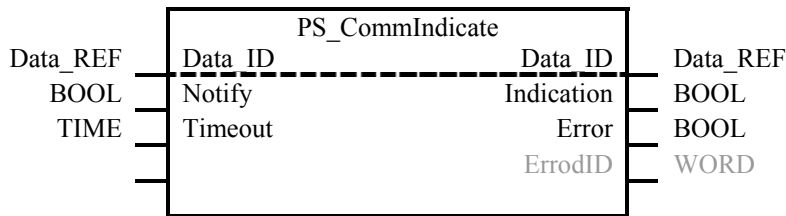
5.2.5. PS_CommNotify

FB-Name		PS_CommNotify	
<ul style="list-style-type: none"> This function block is executed by a server and creates an immediate notification by telegram transmission, creating an unconfirmed and unrequested message to a client. The FB is executed once and does not trigger a response telegram. 			
VAR_IN_OUT			
B	Data_ID	Data_REF	Reference to Data_ID
VAR_INPUT			
B	Notify	BOOL	Triggers a notification telegram, edge-trigger
B	Timeout	TIME	Supervisory timeout
VAR_OUTPUT			
B	Notified	BOOL	True after FB Executed; cleared by Termination or Loss of connection (Timeout)
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing
E	ErrorID	WORD	Error identification
Notes :			



5.2.6. PS_CommIndicate

FB-Name		PS_CommIndicate	
<ul style="list-style-type: none"> This function block is executed by the client and terminates the communication from a CommResponse, handling the received data. The FB is executed once. 			
VAR IN OUT			
B	Data_ID	Data_REF	Reference to Data_ID
VAR INPUT			
B	Notify	BOOL	Enables the FB to receive Notification during execution of block, level triggered
B	Timeout	TIME	Supervisory timeout
VAR OUTPUT			
B	Indication	BOOL	True after receipt of Notification; cleared by Termination of FB execution or Loss of connection (Timeout)
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing
E	ErrorID	WORD	Error identification
Notes :			



5.3. Communication by Publisher/Subscriber

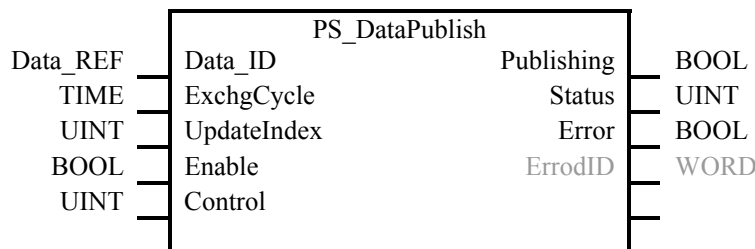
The Publisher/subscriber model allows to exchange data on a network by means of publishing and subscribing to variables. It offers widest flexibility and asynchronous as well as synchronous connection methods independent from network technologies.

The sender publishes variables (data structures) at a certain cycle time without confirmed connections. Subscribers (one or many) may consume the published information at a different cycle time, synchronous or not. A quality variable informs about the age of information in sender cycle counts.

5.3.1. PS_Publish

Publishes a variable or structure on a network. Certain additional “quality of information” meta-data is published additionally to provide information to the subscriber about age and timeliness of data retrieved by subscription.

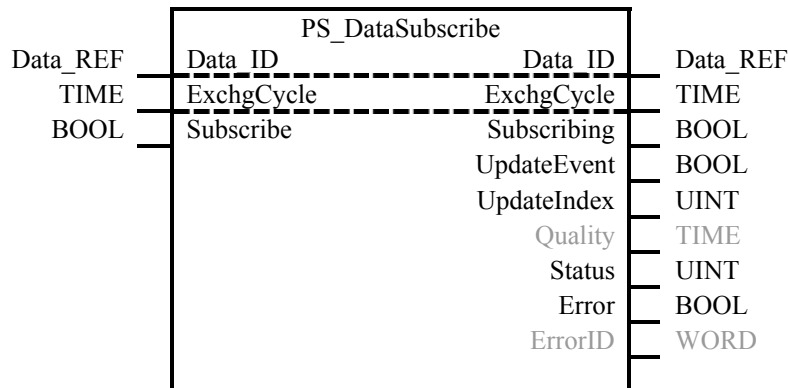
FB-Name		PS_DataPublish	
<ul style="list-style-type: none"> This function block publishes variables (a data structure referenced). User defines a network update time period, the Exchange Cycle Time User increments the cycle index on each data update provided to FB 			
VAR INPUT			
B	Data_ID	Data_REF	Input Reference to Data to be published
B	ExchgCycle	TIME	Input The cycle time of publishing (transmission) cycle
B	UpdateIndex	UINT	Input by user: Counts cyclically from 0 to 65535 continuously as a publishing cycle counter for every update of Data the user provides to the FB for publication. Not a counter to accrued network cycles.
B	Enable	BOOL	Triggers a publishing of a data structure
B	Control	UINT	Control Word for Publication purposes, implementation specific
VAR OUTPUT			
B	Publishing	BOOL	True while FB executes
B	Status	UINT	Status information
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new command while still executing
E	ErrorID	WORD	Error identification
Notes : Status information includes at minimum: 01h: 0: no error, + 1: transmission link status error, 02h: 0: no error, 1: cycle time violation, the transmission rate, protocol and size of data structure do not allow transmission of data within selected Exchange cycle time			



5.3.2. PS_Subscribe

Subscribes to a variable or structure on a network. Certain additional quality of information data is to be retrieved from the transmitted data, e.g. the CycleIndex.

FB-Name		PS_DataSubscribe		
<ul style="list-style-type: none"> This function block subscribes to variables (a data structure referenced). 				
VAR_IN_OUT				
B	Data_ID	Data_REF	Reference to Data to be published	
B	ExchgCycle	TIME	The cycle time of publishing (transmission) cycle	
VAR_INPUT				
B	Subscribe	BOOL	Triggers a subscription of a data structure, level sensitive	
VAR_OUTPUT				
B	Subscribing	BOOL	True while FB executes	
B	UpdateEvent	BOOL	Rising edge on every increment of UpdateIndex as notification to receiver	
B	UpdateIndex	UINT	Output from subscribed Data: counter cycles from 0 to 65535 continuously as a publishing cycle counter for every Data update containing new information the user provides.	
E	Quality	TIME	This variable shows how "old" the value of this subscriber network variable is by a timer implementation in the subscriber. As long as Data is not updated on the network, this value gets increased by 100 µs increments. This could occur e.g. by asynchronous nature of transmission or when the link of the network connection is lost. After re-establishing the connection, the value is set back to 0 again. Max. value of Quality is 65535 sec, then starting at 0.	
B	Status	UINT	Status information if the Data subscription	
B	Error	BOOL	Signals that an error has occurred within Function Block, e.g. new request command while still executing	
E	ErrorID	WORD	Error identification	
<p>Notes : Status information includes at minimum:</p> <p>01h: 0: no error, + 1: transmission link status error,</p> <p>02h: 0: no error, 1: cycle time violation, the transmission rate, protocol and size of data structure do not allow reception of data within selected Exchange cycle time</p> <p>04h: Type or Size mismatch between Data_REF and received data</p> <p>08h: Cycle Time Violation Flag, data transmission is slower than configured publication cycle time.</p>				



Glossary, References and Standards

Glossary

References

Figures

Figure 1: Signal Naming for edge- and level triggered input signals.....	6
Figure 2: PS_wind_csv timing diagram.....	10
Figure 3: PS_Wind_ct timing diagram.....	11
Figure 4: PS_Dancer Control timing diagram.....	13
Figure 5: PS_Dancer Control Structure.....	13
Figure 6: PS_Registration timing diagram.....	15
Figure 7: PS_Reg_Correction timing diagram.....	17
Figure 8: PS_Indexing timing diagram.....	19
Figure 9: Digital PLS timing diagram.....	22
Figure 10: Digital PLS, detailed description of Switch01.....	22
Figure 11: Digital PLS duration switch timing diagram.....	23
Figure 12: PS_SetOverride timing diagram.....	24
Figure 13: PS_Jog Axis timing diagram.....	25
Figure 14: Flying Sync Geometries.....	26
Figure 15: PS_FlyingSync timing diagram for a single synchronization.....	28
Figure 16: PS_GearInDyn structure of gear model.....	29
Figure 17: PS_OrientedStop graphical explanation of behavior.....	31
Figure 18: PackML State Model.....	33
Figure 19: PS_Pack_ML_State_Model FB Names and application adoption.....	34
Figure 20: The OMAC PWG Application Library Logo.....	49

Tables

Table 1: Defined Function Blocks of PackAL.....	8
Table 2: PS_Index Position table.....	18
Table 3: Digital PLS Structure table.....	20
Table 4: Digital PLS Track table.....	21
Table 5: Digital PLS Switch Position table.....	21
Table 6: Supported datatypes.....	46
Table 7: Supported derived datatypes.....	46
Table 8: Short overview of the Function Blocks.....	47

Standards

References to other standards:

IEC 61131-3 2nd edition, 2003

Machine Directive 98/37/EC, clause 1.2.5. - and related US and Asia directives

Function Blocks for Motion Control, V. 1.0, PLCopen, 2001

Function Blocks for Motion Control, Part 2, Extensions, PLCopen, 2005

Appendix A. Compliance Declaration and Compliance List

Listed in this Appendix are the requirements for the compliance statement from the supplier of the Application Library Function Blocks. This part should be seen as integral to the Application Library.

The compliance statement consists of two main groups: supported data types and supported Function Blocks, in combination with the applicable inputs and outputs. The supplier has to fill out the tables for the used data types and Function Blocks, according to their product, committing their support to the specification.

OMAC PWG WILL APPROVE THE SUBMITTED DOCUMENTS FOR FORMAL COMPLETENESS ONLY, AND MAKES NO WARRANTIES OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING ANY WARRANTY TO THE MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, FOR THE APPROVED PRODUCT OR ITS APPROVED COMPLIANCE DOCUMENTATION. IN NO EVENT WILL OMAC BE RESPONSIBLE FOR ANY LOSS OR DAMAGE ARISING OUT OR RESULTING FROM ANY DEFECT, ERROR OR OMISSION IN THE APPROVED PRODUCT COMPLIANCE DOCUMENT OR FROM ANYONE'S USE OF OR RELIANCE ON THIS COMPLIANCE DOCUMENT AND ITS APPROVAL BY OMAC.

By submitting these tables to OMAC PWG, and after approval by OMAC PWG, the list will be published on the OMAC website, www.OMAC.org, as well as a short form overview, as specified here below.

In addition to this approval, the supplier gets access and usage rights of the OMAC PWG logo, as described in Appendix B - The OMAC PWG Logo and Its Usage.

Statement of Supplier

Supplier name	
Supplier address	
City	
Country	
Telephone	
Fax	
Email address	
Product Name	
Product version	
Release date	

I herewith state that the following tables as filled out and submitted do match our product as well as the accompanying user manual, as stated above.

Name of representation (person):

Date of signature (dd/mm/yyyy):

Signature:

Supported Datatypes

Defined datatypes with MC library:	Supported	If not supported, which datatype used
BOOL		
INT		
UINT		
WORD		
REAL		
LREAL		
TIME		
PS Direction		

Table 6: Supported datatypes

Within the specification the following derived datatypes are defined. Which structure is used in this system:

Derived datatypes:	Where used	Supported	Which structure
Axis_REF	Nearly all FBs		
DANCER_REF	PS Dancer_Control		
Input_REF	PS-Registration		
POS_REF	PS Index		
DIGPLS_REF	PS DigitalPLS		
OUTPUT_REF	PS DigitalPLS		
TRACK_REF	PS DigitalPLS		
MACHINE_REF	PS_PackML_State_Machine		
Data_REF	Numerous FBs		
Sender_REF	PS_Send, PS_Receive		
Receiver_REF	PS_Send, PS_Receive		

Table 7: Supported derived datatypes

Datatypes

The data type REAL listed in the Function Blocks and parameters (e.g. for velocity, acceleration, distance, etc.) may be exchanged to SINT, INT, DINT or LREAL without to be seen as incompliant to this standard, as long as they are consistent for the whole set of Function Blocks and parameters.

Implementation allows to extend data types as long as the basic data type is kept. For example: WORD may be changed to DWORD, but not to REAL.

Function Blocks and Inputs and Outputs

An implementation which claims compliance with this specification shall offer a set of Function Blocks for motion control, meaning one or more, with at least the basic input and output variables, marked as “B” in the tables. These inputs and outputs have to be supported to be compliant.

For higher-level systems and future extensions any subset of the extended input and output variables, marked as “E” in the tables can be implemented.

Supplier specific additions are marked with “S”, and can be listed as such in the supplier documentation.

- Basic input/output variables are mandatory Marked in the tables with the letter “B”
- Extended input /output variables are optional Marked in the tables with the letter “E”
- Supplier Specific additions Marked in the supplier’s compliance documentation with “S”

All supplier specific in- and outputs of all FBs must be listed in the certification list of the supplier. With this, the certification listing from a supplier describes all the I/Os of the relevant FBs, including supplier-specific extensions, and thus showing the complete FBs as used by the supplier.

Overview of the Function Blocks

Packaging Process Function Blocks	Supported Yes / No	Comments (<= 48 char.)
PS_Wind_csv		
PS_Wind_ct		
PS_Dancer_Control		
PS_Registration		
PS_Reg_Correction		
PS_Indexing		
PS_Batch_Counter		
PS_DigitalPLS		
PS_SetOverride		
PS_JogAxis		
PS_FlyingSync		
PS_GearInDyn		
PS_OrientedStop		
Machine Behavior Function Blocks	Supported Yes / No	Comments (<= 48 char.)
PS_PackML_State_Machine		
MachineCommunication Function Blocks	Supported Yes / No	Comments (<= 48 char.)
PS_Send,		
PS_Receive		
PS_CommRequest		
PS_Indicate		
PS_CommRespond		
PS_CommConfirm		
PS_Publish		
PS_Subscribe		

Table 8: Short overview of the Function Blocks

FB Conformity Declaration

Supplier to provide a complete table of all Interface Variables per Function Block for all FB's listed in this standard under Appendix A Table 8 including a "Supported - Yes/No" column and a Comment Column.

Example:

If Supported		Yes/No	Comment
PS_Wind_csv			
VAR_IN_OUT			
B	Axis		
VAR_INPUT			
B	Enable		
B	Velocity		
B	SpoolRadius		
B	Min_S_Radius		
B	Max_S_Radius		
E	Direction		
E	Acceleration		
E	Deceleration		
E	Jerk		
VAR_OUTPUT			
B	Busy		
B	InVelocity		
B	Error		
B	ErrorID		

Appendix B: The OMAC PWG Logo and it's usage

For quick identification of compliant products, OMAC PWG has developed a logo for the Application Library Function Blocks:



Figure 20: The OMAC PWG Application Library Logo

This logo is owned and trademarked by OMAC.

In order to use this logo free-of-charge, the relevant company has to fulfill all the following requirements:

1. the company has to be a member of OMAC PWG;
2. the company has to comply to the existing specification, as specified by the OMAC PWG, PackSoft committee, and as published by OMAC, and of which this statement is a part;
3. this compliance is done in written form by the company to OMAC, clearly stating the applicable software package and the supporting elements of all the specified tables, as specified in the document itself;
4. in case of non-fulfillment, which has to be decided by OMAC PWG, the company will receive a statement on this from OMAC PWG in written form. The company will have a one month period to either adopt their software package in such a way that it complies, represented by the issuing of a new compliance statement, or remove all reference to the specification, including the use of the logo, from all their specification, be it technical or promotional material;
5. the logo has to be used as is - meaning the full logo. It may be altered in size as long as the original scale and color setting is kept. Alternatively, a greyscale version may be used.
6. the logo has to be used in the context of packaging and / or motion control.