Function block library

FunctionModules_4

for PLCnext Engineer

Documentation for
PHOENIX CONTACT function blocks
PHOENIX CONTACT GmbH Co. KG
Flachsmarktstrasse 8
D-32825 Blomberg, Germany

This documentation is available in English only.

Table of Contents

1 Installation hint

Please copy the library data to your PLCnext Engineer (former: PC Worx Engineer) working library directory.

If you did not specify a different directory during **PLCnext Engineer** installation the default PLCnext Engineer working library directory is

C:\Users\Public\Documents\PLCnext Engineer\Libraries

2 General information

This function block library provides function blocks for acquisition, open and closed-loop control (drivers for position detection terminals for incremental encoders and terminals with counting function.

3 Change notes

Library version	Library build	PLCnext Engineer version	Change notes
4	20220316	2021.0 LTS	Examples and documentation improved.
4	20201204	2020.0 LTS	FUM_IL_PWM2: • Bug fix for: xActive = FALSE in case of error • Bug fix for: udiActPosition shows wrong value • Bug fix for: xDirection is always FALSE FUM_IL_PWM2_Data: • Bug fix for: xError = FALSE for invalid iChannel value
3	20201111	2020.0 LTS	FUM_IL_CNT*: Added examples • FUM_3_EXA_C_COUNT.pcwex • FUM_3_EXA_C_FREQ.pcwex • FUM_3_EXA_C_PULSE.pcwex • FUM_3_EXA_C_TIME.pcwex
3	20200206	2020.0 LTS	Released for 2020.0 LTS
3	20191015	2019.0 LTS 2019.3 2019.6 2019.9	New function blocks: • FUM_INC_IN • FUM_INC_DATA • FUM_INC_PARA
2	20191001	2019.0 LTS 2019.3 2019.6 2019.9	Adapted to 2019.9
1	20190716	2019.0 LTS 2019.3 2019.6	Converted from PC Worx 6

New version number: Functional changes of at least one function block, incompatibilities (e.g. change of library format)

New build number: No functional changes, but changes in the ZIP file (e.g. documentation update, additional examples)

4 Supported PLCs

AXC F 1152 (1151412) from version 3, build 20200206
AXC F 2152 (2404267)
AXC F 3152 (1069208) from version 3, build 20201111

5 Function blocks

Function block	Description	Version	Supported articles	License
FUM_C_COUNT	Function block for parameterization of the IB IL CNT-PAC (2861852) terminal in event counting operating mode.	1	IB IL CNT (2836337) IB IL CNT-PAC (2861852) IB IL CNT-2MBD (2855813) IB IL CNT-2MBD-PAC (2862071)	none
FUM_C_FREQ	Function block for parameterization of the IB IL CNT-PAC (2861852) terminal in frequency measurement operating mode and cyclic reading out of the measured values.	1	IB IL CNT (2836337) IB IL CNT-PAC (2861852) IB IL CNT-2MBD (2855813) IB IL CNT-2MBD-PAC (2862071)	none
FUM_C_PULSE	Function block for parameterization of the IB IL CNT-PAC (2861852) terminal in pulse generator operating mode.	1	IB IL CNT (2836337) IB IL CNT-PAC (2861852) IB IL CNT-2MBD (2855813) IB IL CNT-2MBD-PAC (2862071)	none
FUM_C_TIME	Function block for parameterization of the IB IL CNT-PAC (2861852) terminal in time measurement operating mode and cyclic reading out of the measured values.	1	IB IL CNT (2836337) IB IL CNT-PAC (2861852) IB IL CNT-2MBD (2855813) IB IL CNT-2MBD-PAC (2862071)	none

FUM_IL_PWM2	Function block for the communication between PLC and IB IL PWM/2 terminal.	2	IB IL PWM/2 (2742612) IB IL PWM/2-PAC (2861632)	none
FUM_IL_PWM2_Para	Function block for parameterization and scaling of four operating modes of the IB IL PWM/2-PAC (2861632) terminal.	1	IB IL PWM/2 (2742612) IB IL PWM/2-PAC (2861632)	none
FUM_IL_PWM2_Data	Function block for diagnosis information from the IB IL PWM/2-PAC (2861632) terminal.	2	IB IL PWM/2 (2742612) IB IL PWM/2-PAC (2861632)	none
FUM_INC_IN	Function block for parameterization and scaling of position values of the IB IL INC (2836324) terminal.	1	IB IL INC (2836324) IB IL INC-PAC (2861849) IB IL INC-2MBD (2819778) IB IL INC-2MBD-PAC (2819765)	none
FUM_INC_DATA	Auxiliary block for assigning structure variables.	1	IB IL INC (2836324) IB IL INC-PAC (2861849) IB IL INC-2MBD (2819778) IB IL INC-2MBD-PAC (2819765)	none
FUM_INC_PARA	Auxiliary block for assigning structure variables.	1	IB IL INC (2836324) IB IL INC-PAC (2861849) IB IL INC-2MBD (2819778) IB IL INC-2MBD-PAC (2819765)	none

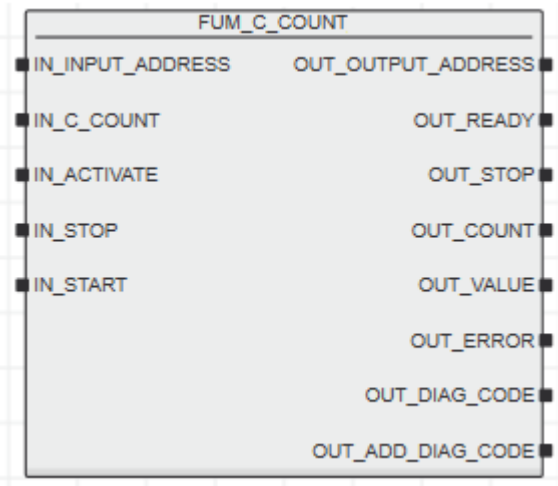
6 FUM_IL_CNT

The FUM_IL_CNT function block group includes the FUM_C_COUNT, FUM_C_FREQ, FUM_C_PULSE and FUM_C_TIME function blocks for parameterization of the IB IL CNT-PAC (2861852) module.

6.1 FUM_C_COUNT

The FUM_C_COUNT function block parameterizes the IB IL CNT-PAC (2861852) module in “event counting” mode and cyclically reads the count value. The module is parameterized with the function block parameters, in this case with a data structure.

6.1.1 Function block call



6.1.2 Operating mode

The FUM_C_COUNT function block only supports “event counting” mode. The function block parameters (in the data structure) must be supplied with valid data in order for the operating mode to be executed successfully. Start this mode with a positive edge at the IN_ACTIVATE input. A rising edge at the OUT_READY output indicates successful completion of parameterization, i.e., valid count values are now available at the OUT_VALUE output. If the OUT_ERROR output parameter is set (group error message), the diagnostics must be evaluated. Once the error has been removed, a new positive edge at the IN_ACTIVATE input triggers a new parameterization process and deletes the error. The IN_STOP input can interrupt event counting at any time. The positive edge at the OUT_STOP output indicates that the counter has been stopped. The current count value can now be read from the OUT_VALUE output. The interrupted counting is restarted by a positive edge at the IN_START input. A positive edge at the OUT_READY output indicates that the counter is active again.

6.1.3 Input parameters

Name	Type	Description
IN_INPUT_ADDRESS	DWORD	Process data input address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
IN_C_COUNT	STR_C_COUNT	Data structure with the parameters for the IB IL CNT-PAC (2861852) module.
IN_ACTIVATE	BOOL	A positive edge starts the parameterization of the module and the subsequent reading of measured values.

IN_STOP	BOOL	A positive edge stops “event counting” mode. A positive edge at the OUT_STOP output indicates that the operating mode has been stopped.
IN_START	BOOL	A positive edge restarts “event counting” mode. A positive edge at the OUT_READY output indicates that event counting has started.

6.1.4 STR_C_COUNT data structure

Name	Type	Description
I_BehaviourBusReset	BOOL	Behavior in the event of an INTERBUS reset.
I_InputSwitching	INT	Possible input wiring (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0 dec to 3dec
I_SourceGateConnection	INT	Possible input links (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0 dec to 15dec
I_Gate	INT	Gate input condition that must be met for the counting process (see section 4.3 “Event Counting Mode” in the User Manual). Value range: 0 dec to 7dec
I_Continuous	BOOL	Counting repeat FALSE: Single count TRUE: Continuous count
I_Direction	BOOL	Counting direction FALSE: Down TRUE: Up
I_Output	INT	Switching behavior of the digital output when the final value is reached (see section 4.3 “Event Counting Mode” in the User Manual). Value range: 0 dec to 7dec
I_PulseWidth	INT	Possible times for the duration of the output pulse (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0 dec to 7dec
I_StartValue	DINT	Start value: Value range: 0 to 16777215
I_EndValue	DINT	Final value: Value range: 0 to 16777215

6.1.4.1 Assigning structure variables

Another function block (PC_COUNT) is used to assign the variables in the data structure. This function block provides the exact variables of the “STR_C_COUNT” data structure as inputs. The “OUT_C_COUNT” output of the PC_COUNT function block can be directly linked to the “IN_C_COUNT” input of the C_COUNT function block.

6.1.5 Output parameters

Name	Type	Description
OUT_OUTPUT_ADDRESS	DWORD	Process data output address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
OUT_READY	BOOL	A rising edge indicates that the module has been parameterized and event counting has been started. Valid values are now present at the OUT_VALUE output.
OUT_STOP	BOOL	A rising edge indicates that the counter has been stopped. The current counter value is output at the OUT_VALUE output.
OUT_COUNT	BOOL	Count value reached.
OUT_VALUE	DINT	Count value of counter (24-bit resolution).
OUT_ERROR	BOOL	A rising edge indicates an error message. TRUE: Error detected, evaluation of OUT_DIAG_CODE and OUT_ADD_DIAG_CODE. FALSE: No error detected.
OUT_DIAG_CODE	WORD	Diagnostic code for the error message.
OUT_ADD_DIAG_CODE	WORD	Additional diagnostic code for the error message.

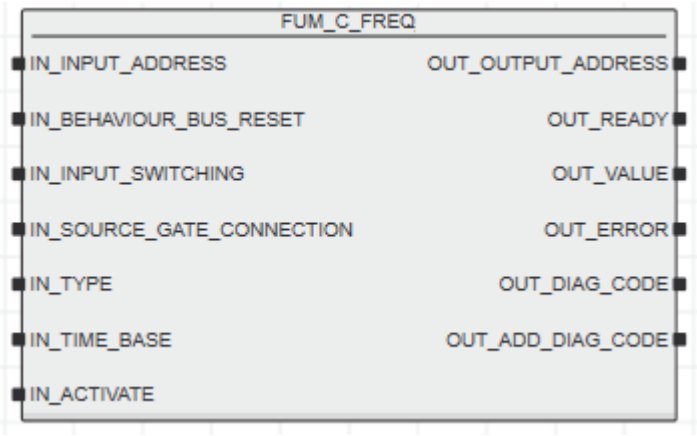
6.1.6 Diagnosis

wDiagCode	wAddDiagCode	Description
16#FF01	16#0000	Parameterization error.
	16#0001	Timeout, module is not responding.
	16#0002	“I_InputSwitching” parameter invalid.
	16#0003	“I_SourceGateConnection” parameter invalid.
	16#0004	“I_Gate” parameter invalid.
	16#0005	“I_Output” parameter invalid.
	16#0006	“I_PulseWidth” parameter invalid.
	16#0007	“I_StartValue” parameter invalid.
	16#0008	“I_EndValue” parameter invalid.
16#FF02	16#0000	Terminal error message.
	16#0000	Terminal error message. • Invalid parameters in an operating mode specification • Counter read when not in specified operating mode • A set reserved bit • Counter started when not in specified operating mode

6.2 FUM_C_FREQ

The FUM_C_FREQ function block parameterizes the IB IL CNT-PAC (2861852) module in “frequency measurement” mode and cyclically reads the measured values. The module is parameterized with the block parameters.

6.2.1 Function block call



6.2.2 Operating mode

The FUM_C_FREQ function block only supports “frequency measurement” mode. The block parameters must be supplied with valid data in order for the operating mode to be executed successfully. Start this mode with a positive edge at the IN_ACTIVATE input. A rising edge at the OUT_READY output indicates successful completion of parameterization, i.e., valid measured values are now available at the OUT_VALUE output. If the OUT_ERROR output parameter is set (group error message, see 0), the diagnostics must be evaluated. Once the error has been removed, a new positive edge at the IN_ACTIVATE input triggers a new parameterization process and deletes the error.

6.2.3 Input parameters

Name	Type	Description
IN_INPUT_ADDRESS	DWORD	Process data input address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
IN_BEHAVIOUR_BUS_RESET	BOOL	Behavior in the event of an INTERBUS reset: FALSE: Pulse mode is reset in the event of an INTERBUS reset. TRUE: No response in the event of an INTERBUS reset
IN_INPUT_SWITCHING	INT	Possible input wiring (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0dec to 3dec
IN_SOURCE_GATE_CONNECTION	INT	Possible input links (see section 4.6 “System Setting” Command" in the User Manual) Value range: 0dec to 15dec.
IN_TYPE	INT	Frequency measurement options: 1 = Time-controlled time values, see IN_TIME_BASE parameter 2 = State-controlled (high level) 3 = State-controlled (low level) 4 = State-controlled (rising edge) 5 = State-controlled (falling edge), (see section 4.2 “Frequency Measurement Mode” in the User Manual)
IN_TIME_BASE	INT	Gate time for frequency measurement. Possible values in ms: 1, 2, 10, 50, 100, 500, 1000
IN_ACTIVATE	BOOL	A positive edge starts the parameterization of the module and the subsequent reading of measured values.

6.2.4 Output parameters

Name	Type	Description
OUT_OUTPUT_ADDRESS	DWORD	Process data output address of the IB IL CNT-PAC (2861852).
OUT_READY	BOOL	A rising edge indicates that the module has been parameterized and event counting has been started. Valid values are now present at the OUT_VALUE output.
OUT_VALUE	REAL	In “time-controlled” mode the standardized frequency is output in Hz. Caution: The values are only valid once the set gate time has expired. In “state-controlled” mode the counted pulses of the relevant measurement are output.
OUT_ERROR	BOOL	A rising edge indicates an error message. TRUE: Error detected, evaluation of OUT_DIAG_CODE and OUT_ADD_DIAG_CODE. FALSE: No error detected.
OUT_DIAG_CODE	WORD	Diagnostic code for the error message.
OUT_ADD_DIAG_CODE	WORD	Additional diagnostic code for the error message.

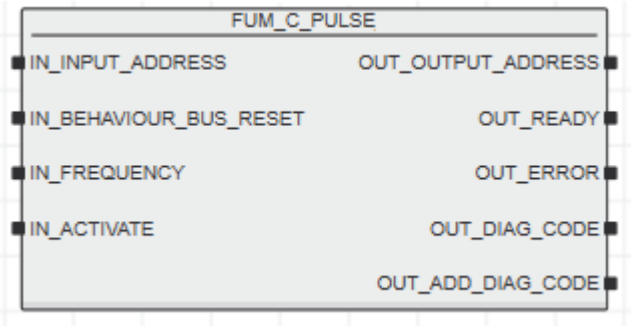
6.2.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#FF01	16#0000	Parameterization error.
	16#0001	Timeout, module is not responding.
	16#0002	“IN_InputSwitching” parameter invalid.
	16#0003	“IN_SourceGateConnection” parameter invalid.
	16#0004	"“IN_TYPE” parameter invalid.
	16#0005	“IN_TIME_BASE” parameter invalid.
16#FF02	16#0000	Terminal error message.
	16#0000	Terminal error message. • Invalid parameters in an operating mode specification • Counter read when not in specified operating mode • A set reserved bit • Counter started when not in specified operating mode

6.3 FUM_C_PULSE

The FUM_C_PULSE function block parameterizes the IB IL CNT-PAC (2861852) module in “pulse generator” mode. The module is parameterized with the block parameters.

6.3.1 Function block call



6.3.2 Operating mode

The FUM_C_PULSE function block only supports “pulse generator” mode. The block parameters must be supplied with valid data in order for the operating mode to be executed successfully. Start this mode with a positive edge at the IN_ACTIVATE input. A rising edge at the OUT_READY output indicates successful completion of parameterization, i.e., pulse output mode is now started. If the OUT_ERROR output parameter is set, the diagnostics must be evaluated. Once the error has been removed, a new positive edge at the IN_ACTIVATE input triggers a new parameterization process and deletes the error.

6.3.3 Input parameters

Name	Type	Description
IN_INPUT_ADDRESS	DWORD	Process data input address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
IN_BEHAVIOUR_BUS_RESET	BOOL	Behavior in the event of an INTERBUS reset: FALSE: Pulse mode is reset in the event of an INTERBUS reset. TRUE: No response in the event of an INTERBUS reset
IN_FREQUENCY	INT	Target frequency for “pulse generator” mode. The frequency can be specified in 500 Hz increments in the designated range. Value range: 1000 Hz to 10000 Hz (See section 4.5 “Pulse Generator Mode” in the User Manual).
IN_ACTIVATE	BOOL	A positive edge starts the parameterization of the module and the subsequent output of pulses.

6.3.4 Output parameters

Name	Type	Description
OUT_OUTPUT_ADDRESS	DWORD	Process data output address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
OUT_READY	BOOL	A rising edge indicates that the module has been parameterized and pulse output has been started.
OUT_ERROR	BOOL	A rising edge indicates an error message. TRUE: Error detected, evaluation of OUT_DIAG_CODE and OUT_ADD_DIAG_CODE. FALSE: No error detected.
OUT_DIAG_CODE	WORD	Diagnostic code for the error message.
OUT_ADD_DIAG_CODE	WORD	Additional diagnostic code for the error message.

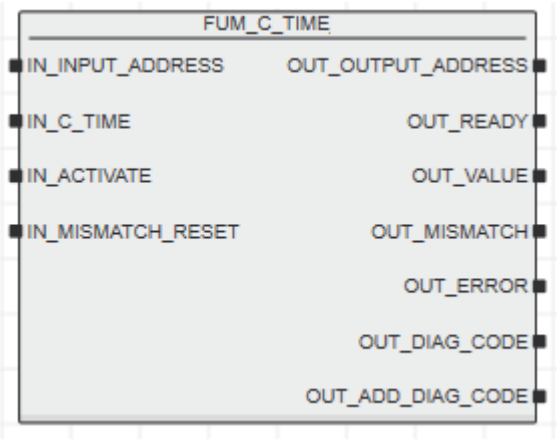
6.3.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#FF01	16#0000	Parameterization error.
	16#0001	Timeout, module is not responding.
	16#0002	“IN_FREQUENCY” parameter invalid.
16#FF02	16#0000	Terminal error message.
	16#0000	Terminal error message. • Invalid parameters in an operating mode specification • Counter read when not in specified operating mode • A set reserved bit • Counter started when not in specified operating mode

6.4 FUM_C_TIME

The FUM_C_TIME function block parameterizes the IB IL CNT-PAC (2861852) module in “time measurement” mode and cyclically reads the measured values. The module is parameterized with the block parameters, in this case with a data structure.

6.4.1 Function block call



6.4.2 Operating mode

The FUM_C_TIME function block only supports “time measurement” mode. The block parameters (in the data structure) must be supplied with valid data in order for the operating mode to be executed successfully. Start this mode with a positive edge at the IN_ACTIVATE input. A rising edge at the OUT_READY output indicates successful completion of parameterization, i.e., valid measured values are now available at the OUT_VALUE output. If the OUT_ERROR output parameter is set, the diagnostics must be evaluated. Once the error has been removed, a new positive edge at the IN_ACTIVATE input triggers a new parameterization process and deletes the error.

6.4.3 Input parameters

Name	Type	Description
IN_INPUT_ADDRESS	DWORD	Process data input address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
IN_C_TIME	STR_C_TIME	Data structure with the parameters for the IB IL CNT-PAC (2861852) module.
IN_ACTIVATE	BOOL	A positive edge starts the parameterization of the module and the subsequent output of pulses.
IN_MISMATCH_RESET	BOOL	Setting this input resets the mismatch message (only in conjunction with I_BehaviourOutput = FALSE). If the OUT_MISMATCH output is FALSE, this input can also be reset.

6.4.4 STR_C_TIME data structure

Name	Type	Description
I_BehaviourBusReset	BOOL	Behavior in the event of an INTERBUS reset.
I_InputSwitching	INT	Possible input wiring (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0dec to 3dec
I_SourceGateConnection	INT	Possible input links (see section 4.6 “System Setting” Command" in the User Manual). Value range: 0dec to 15dec
I_Resolution	INT	Resolution value of the LSB (see section 4.4 “Time Measurement Mode” in the User Manual). Values: 2 µs => 2dec, 2 ms => 2000dec,10 ms => 10000dec
I_BehaviourOutput	BOOL	Switching behavior of the digital output<br< FALSE: The digital output is not used, message sent via the “OUT_MISMATCH” parameter, which must be acknowledged. TRUE: The digital output indicates that the values are outside a specified range. This is indicated in parallel via the “OUT_MISMATCH” parameter.
I_MeasurementType	BOOL	Type of measurement FALSE: Measurement of period length TRUE: Measurement of pulse length
I_Mismatch	INT	Display behavior of limit values (see section 4.4 “Time Measurement Mode” in the User Manual). Value range: 0dec to 7dec
I_StartValue	DINT	Standardized start value, depending on the “I_Resolution” parameter. Value range: at 2 µs/ms, 0 to 131070 µs/ms and at 10 ms, 0 to 655350 ms
I_EndValue	DINT	Standardized final value, depending on the “I_Resolution” parameter. Value range: at 2 µs/ms, 0 to 131070 µs/ms and at 10 ms, 0 to 655350 ms

6.4.4.1 Assigning structure variables

Another function block (PC_TIME) is used to assign the variables in the data structure. This function block provides the exact variables of the “STR_C_TIME” data structure as inputs. The “OUT_C_TIME” output of the PC_TIME function block can be directly linked to the “IN_C_TIME” input of the C_TIME function block.

6.4.5 Output parameters

Name	Type	Description
OUT_OUTPUT_ADDRESS	DWORD	Process data output address of the IB IL CNT-PAC (2861852) module (ensure data consistency).
OUT_READY	BOOL	A rising edge indicates that the module has been parameterized and event counting has been started. Valid values are now present at the OUT_VALUE output.
OUT_VALUE	DINT	Measured value of the parameterized time measurement. The resolution depends on the “I_Resolution” parameter. Value range: at 2 µs/ms, 0 to 131070 µs/ms and at 10 ms, 0 to 655350 ms.
OUT_MISMATCH	BOOL	Mismatch display, behavior depends on the “I_BehaviourOutput” parameter.
OUT_ERROR	BOOL	A rising edge indicates an error message. TRUE: Error detected, evaluation of OUT_DIAG_CODE and OUT_ADD_DIAG_CODE. FALSE: No error detected.
OUT_DIAG_CODE	WORD	Diagnostic code for the error message.
OUT_ADD_DIAG_CODE	WORD	Additional diagnostic code for the error message.

6.4.6 Diagnosis

wDiagCode	wAddDiagCode	Description
16#FF01	16#0000	Parameterization error.
	16#0001	Timeout, module is not responding.
	16#0002	“I_InputSwitching” parameter invalid.
	16#0003	“I_SourceGateConnection” parameter invalid.
	16#0004	“I_Resolution” parameter invalid.
	16#0005	“I_Mismatch” parameter invalid.
	16#0007	“I_StartValue” parameter invalid.
	16#0007	“I_EndValue” parameter invalid.
16#FF02	16#0000	Terminal error message.
	16#0000	Terminal error message. • Invalid parameters in an operating mode specification • Counter read when not in specified operating mode • A set reserved bit • Counter started when not in specified operating mode

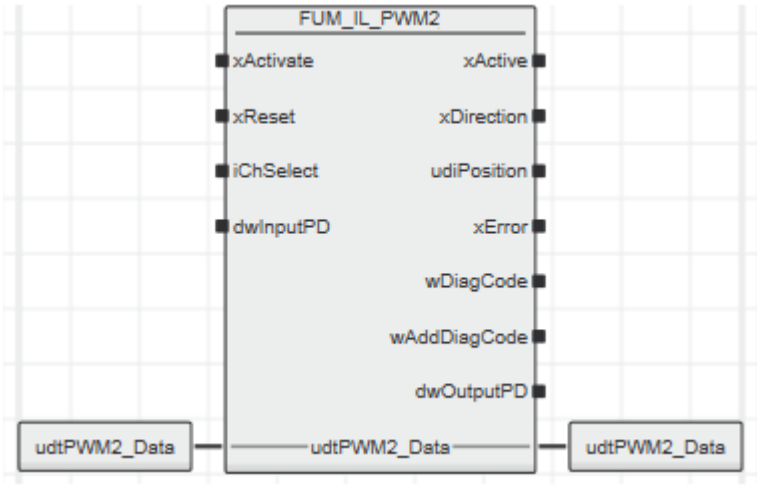
7 FUM_IL_PWM2

The FUM_IL_PWM2 function block group includes the FUM_IL_PWM2, FUM_IL_PWM2_Para and FUM_IL_PWM2_Data function blocks for parameterization and communications with the IB IL PWM/2-PAC (2861632) module.

7.1 FUM_IL_PWM2

This is the central function block for communicating with an IB IL PWM/2-PAC (2861632) module. The function block receives parameters of the FUM_IL_PWM2_Para block via the udtPWM2_Data exchange structure and forwards diagnostic information to the FUM_IL_PWM2_Data function block. It communicates directly with the module via process data.

7.1.1 Function block call



7.1.2 Input parameters

Name	Type	Description
xActivate	BOOL	Rising edge: Activates the function block. FALSE: Deactivates the function block.
xReset	BOOL	Rising edge: Resets the function block.
iChSelect	INT	Selection of the channels to be controlled 0 = Channel 1 and channel 2 1 = Channel 1 2 = Channel 2
dwInputPD	DWORD	IN process data.

7.1.3 Output parameters

Name	Type	Description
xActive	BOOL	FALSE: Function block is not active. TRUE: Function block is active. Do not start any further action unless xActive is TRUE after activation!
xDirection	BOOL	Displays the direction of rotation in pulse direction mode. • FALSE: reverse running • TRUE: forward running
udiPosition	UDINT	Displays the current position in pulse direction mode.
xError	BOOL	TRUE: An error has occurred. For more details refer to wDiagCode and wAddDiagCode.
wDiagCode	WORD	Diagnosis code. Refer to diagnostic table.
wAddDiagCode	WORD	Additional diagnosis code. Refer to diagnostic table.
dwOutputPD	DWORD	OUT process data.

7.1.4 Inout parameters

Name	Type	Description
udtPWM2_Data	FUM_UDT_PWM2_DATA	Exchange structure for other PWM blocks.

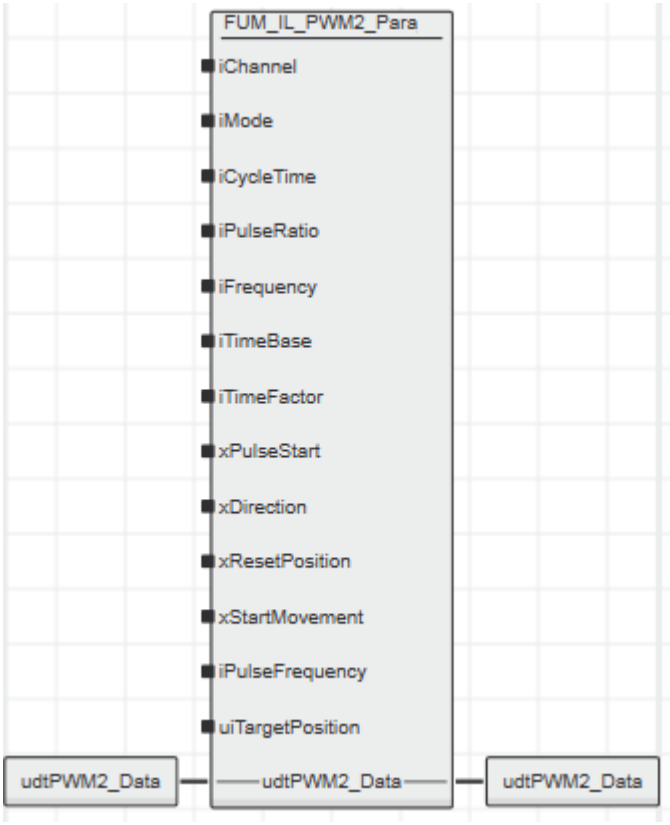
7.1.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8000	16#0000	Function block is in regular operation.
16#C110	16#0000	Invalid iMode parameter at channel 1 on the FUM_PWM2_Para function block.
	16#0100	Value > 4 or < 1.
16#C120	16#0000	Invalid iMode parameter at channel 2 on the FUM_PWM2_Para function block
	16#0100	Value > 4 or < 1.
16#C130	16#0000	Invalid iChSelect parameter on the FUM_IL_PWM2 function block.
	16#0100	Value > 2 or < 0
16#C140	16#0000	Invalid iChannel parameter on the FUM_PWM2_Para block.
	16#0100	Value > 2 or < 1.

7.2 FUM_IL_PWM2_Para

The operating mode and the parameters required for the operating mode are set on this function block. These are then transmitted via the udtPWM2_Data exchange structure to FUM_IL_PWM2 function block. Please note that the function block must be called once per channel. This means that if both channels are required, the block must be called twice (see Startup instructions).

7.2.1 Function block call



7.2.2 Input parameters

Name	Type	Description
iChannel	INT	Select to which channel the parameters should be transmitted. Value range 1 - 2: 1 = Transmit parameter to channel 1 2 = Transmit parameter to channel 2
iMode	INT	Terminal block operating mode for one channel. Value range 1 - 4: 1 = PWM (pulse width modulation) 2 = Frequency generator 3 = Single shot (single pulse generator) 4 = Pulse direction signal
iCycleTime	INT	PWM mode: Values for the frequency specification: 0 = 100 µs 1 = 200 µs 2 = 400 µs 3 = 600 µs 4 = 800 µs 5 = 1 ms 6 = 2 ms 7 = 4 ms 8 = 6 ms 9 = 8 ms 10 = 10 ms 11 = 20 ms 12 = 40 ms 13 = 60 ms 14 = 80 ms 15 = 100 ms 16 = 200 ms 17 = 400 ms 18 = 600 ms 19 = 800 ms 20 = 1 s 21 = 2 s 22 = 4 s 23 = 6 s 24 = 8 s 25 = 10 s
iPulseRatio	INT	Channel pulse ratio (PWM mode); 0 < iPulseRatio < 100
iFrequency	INT	Frequency mode: Output frequency of the terminal block. The value range can be set from 0 to 4095. The minimum resolution of the terminal block is 12.21 Hz per LSB.
iTimeBase	INT	Single shot mode: Time scale for pulse length: 0 = 10 µs (only possible at 5 V output voltage) 1 = 100 µs 2 = 1 ms 3 = 10 ms 4 = 100 ms

iTimeFactor	INT	Single shot mode: Pulse length depending on iTimeBase: Adjustable from 0 - 255. Example for pulse duration of 250 ms: iTimeBase = 3 iTimeFactor = 25
xPulseStart	BOOL	Single shot mode: A positive edge starts the pulse output of the terminal block.
xDirection	BOOL	Pulse direction mode: Adjustment of the direction of rotation. FALSE = reverse running TRUE = forward running
xResetPosition	BOOL	Pulse direction mode: A positive edge sets the positioning counter to 0. Setting only possible via channel 1.
xStartMovement	BOOL	Pulse direction mode: Activates the output with the set control frequency. Setting only possible via channel 1.
iPulseFrequency	INT	Pulse direction mode: Control frequency. Value range 0 - 25 kHz. Setting only possible via channel 1.
uiTargetPosition	UINT	Pulse direction mode: Target position of the motor. Value range 0 - 65535 Setting only possible via channel 1. Note: If specified as value 65535, the motor rotates continuously in the set direction. In doing so, the motor can only be stopped if xStartMovement is set to FALSE.

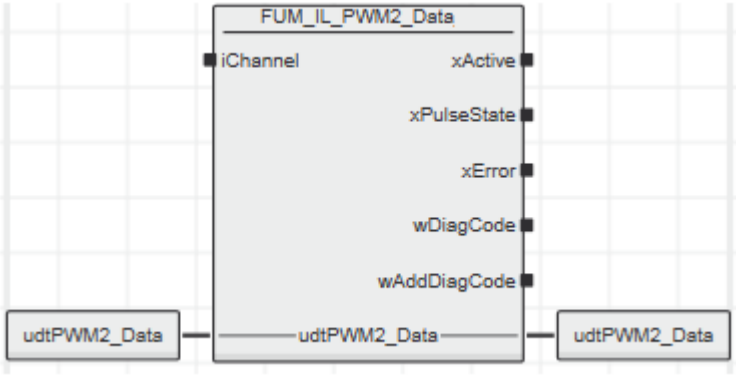
7.2.3 Inout parameters

Name	Type	Description
udtPWM2_Data	FUM_UDT_PWM2_DATA	Exchange structure for other PWM function blocks.

7.3 FUM_IL_PWM2_Data

This function block displays the current status and diagnostic messages of the selected channel. It obtains information by means of the exchange structure udtPWM2_Data. Please note that the function block must be called once per channel. This means that if both channels are required, the function block must be called twice (see Startup instructions).

7.3.1 Function block call



7.3.2 Input parameters

Name	Type	Description
iChannel	INT	Select to which channel the parameters should be transmitted. Value range 1 - 2: 1 = Transmit parameter to channel 1 2 = Transmit parameter to channel 2

7.3.3 Output parameters

Name	Type	Description
xActive	BOOL	FALSE: Function block is not active. TRUE: Function block is active. Do not start any further action unless xActive is TRUE after activation!
xPulseState	BOOL	Only valid in single shot operating mode FALSE = Output pulse is not active TRUE = Output pulse is active
xError	BOOL	TRUE: An error has occurred. For more details refer to wDiagCode and wAddDiagCode.
wDiagCode	WORD	Diagnosis code. Refer to diagnostic table.
wAddDiagCode	WORD	Additional diagnosis code. Refer to diagnostic table.

7.3.4 Inout parameters

Name	Type	Description
udtPWM2_Data	FUM_UDT_PWM2_DATA	Exchange structure for other PWM function blocks.

7.3.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#0000	16#0000	Function block is deactivated.
16#8000	16#0000	Function block is in regular operation.
16#C110	16#0000	Invalid parameterization on FUM_IL_PWM2_Para function block.
	16#0100	iCycleTime parameter invalid. Value > 25 or < 0.
	16#0200	iPulseRatio parameter invalid. Value > 100 or < 0.
	16#0300	Invalid iFrequency parameter. Value > 4095 or < 0.
	16#0400	Invalid iTimeBase parameter. Value > 4 or < 0.
	16#0500	Invalid iTimeFactor parameter. Value > 255 or < 0.
	16#0600	Invalid iPulseFrequency parameter. Value > 25000 or < 0.
16#C120	16#0000	Invalid parameterization on FUM_IL_PWM2_Data function block.
	16#0100	Invalid iChannel parameter on FUM_PWM2_Data block. Value greater than 2 or lower than 1.

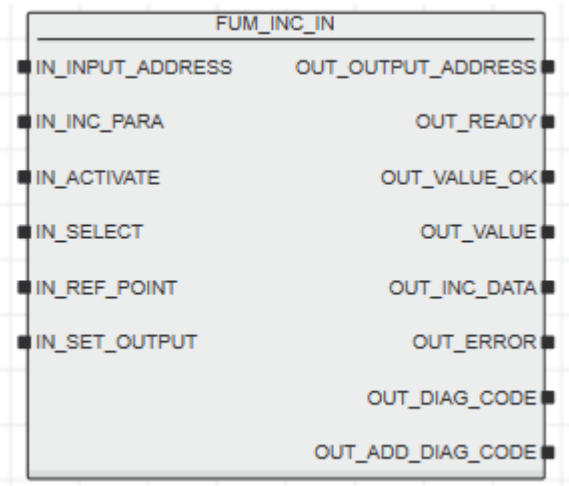
8 FUM_INC_*

The FUM_INC_* function block group includes the FUM_INC_IN, FUM_INC_PARA and FUM_INC_DATA function blocks for parameterization and communications with the IB IL INC-PAC (2861849) module.

8.1 FUM_INC_IN

The FUM_INC_IN function block is programmed to parameterize and scale the position values of the IB IL INC-PAC (2861849) module. This function block scales all the necessary parameters and measured values in a user-specified unit of measurement.

8.1.1 Function block call



8.1.2 Operating modes

The FUM_INC_IN function block supports two operating modes, “ReadPos” (for reading and scaling the position values of the module) and “Reference” (6 versions for homing the module).
“ReadPos” mode starts with a “0” at the IN_SELECT input and a positive edge at the IN_ACTIVATE input. When starting this mode, the IN_INC_PARA parameter (T_INC_IN_PARA data structure) for this mode is automatically activated. Modifying the parameter values of the data structure no longer affects the function of the module. Modified parameters are only accepted on a new positive edge at the IN_ACTIVATE input. The positive edge at OUT_READY and OUT_VALUE_OK indicates that the function block is ready to operate. Valid, scaled measured values are then output at the OUT_VALUE output. Additional status information for the module is provided at the OUT_INC_DATA parameter (T_INC_IN_DATA data structure) of the function block.
“Reference” mode starts when the function block is activated (IN_ACTIVATE = TRUE) by selecting a homing version at the IN_SELECT input. The read position values are not valid during homing. The OUT_VALUE_OK output parameter is thus reset. Successful homing of the module is indicated by a positive edge at the OUT_INC_DATA.O_RefMarkFound output. The IN_SELECT parameter can now be deactivated again and valid positions can be read again with the positive edge of the OUT_VALUE_OK parameter.
If an error message is generated in either operating mode, the error can be identified and removed using the OUT_DIAG_CODE and OUT_ADD_DIAG_CODE error codes. The error can then be acknowledged with a positive edge at the IN_ACTIVATE input.

8.1.3 Input parameters

Name	Type	Description
IN_INPUT_ADDRESS	DWORD	Process data input address of the IB IL INC-PAC (2861849) module (ensure data consistency).
IN_INC_PARA	T_INC_IN_PARA	Data structure with parameters for the incremental encoder.
IN_ACTIVATE	BOOL	A rising edge activates the block and sets the specified parameters (IN_INC_PARA) on the module. Modifying the parameters no longer affects the module settings or the scaling new or modified parameters are only accepted on an edge change from FALSE => TRUE.
IN_SELECT	INT	Selecting the different operating modes: 0 = Read position values. 1 = Homing, set counter to reference point value. 2 = Homing on the next positive edge of one of the digital inputs 3 = Homing on the next negative edge of one of the digital inputs. 4 = Homing on the Z pulse after positive edge of one of the digital inputs. 5 = Homing on the Z pulse after negative edge of one of the digital inputs. 6 = Homing distance encoded incremental encoders. ELGO version. For additional information about the versions, please refer to the data sheet.
IN_REF_POINT	DINT	Reference point value. This value is initialized as a new count value during homing.
IN_SET_OUTPUT	BOOL	This input controls the digital output of the module.

8.1.4 Output parameters

Name	Type	Description
OUT_OUTPUT_ADDRESS	DWORD	Process data output address of the IB IL INC-PAC (2861849) module (ensure data consistency)
OUT_READY	BOOL	A rising edge indicates that the module is ready to operate.
OUT_VALUE_OK	BOOL	A rising edge indicates that the scaled position, which is output at the OUT_VALUE parameter, is valid.
OUT_VALUE	DINT	This value contains the scaled position in “ReadPos” mode (IN_SELECT = 0). This value is only valid if the OUT_VALUE_OK parameter = TRUE.
OUT_INC_DATA	T_INC_IN_DATA	Data structure with the parameters for the IB IL INC-PAC (2861849) module.
OUT_ERROR	BOOL	A rising edge indicates an error message. TRUE => Error detected, evaluation of OUT_DIAG_CODE and OUT_ADD_DIAG_CODE FALSE => No error detected
OUT_DIAG_CODE	WORD	Diagnostic code for the error message.
OUT_ADD_DIAG_CODE	WORD	Additional diagnostic code for the error message.

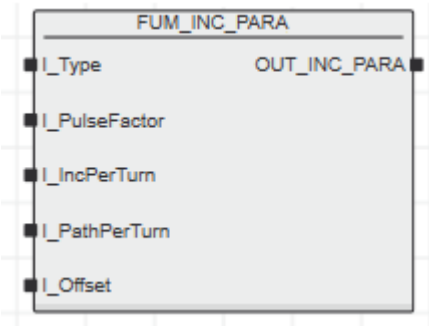
8.1.5 Diagnosis

wDiagCode	wAddDiagCode	Description
16#FF01	16#0000	Parameterization error.
	16#0001	Timeout, module is not responding.
	16#0002	The I_PulsFactor parameter is invalid.
	16#0003	The I_IncPerTurn parameter is invalid.
	16#0004	The I_PathPerTurn parameter is invalid (≤ 0).
	16#0005	The I_PathPerTurn parameter is invalid (> encoder resolution). (Encoder resolution = I_IncPerTurn * I_PulseFactor)
	16#0006	The IN_REF_POINT parameter is smaller than the measuring range.
	16#0007	The IN_REF_POINT parameter is larger than the measuring range.
	16#0008	The IN_SELECT parameter is invalid.
16#FF02	16#0000	Module error message.
	16#0000	Actual position value invalid (voltage supply).
	16#0001	Actual position value invalid (encoder error).
	16#001x	Timeout with error message (x = error code of the module).

8.2 FUM_INC_PARA

The FUM_INC_PARA function block is used to assign the variables in the T_INC_IN_PARA data structure. This function block provides the exact variables of the data structure as inputs. The OUT_INC_PARA output of this function block can be linked directly to the IN_INC_PARA input of the INC_IN function block.

8.2.1 Function block call



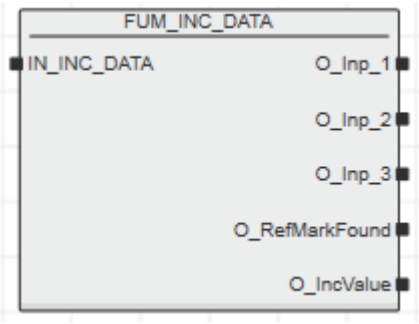
8.2.2 T_INC_IN_PARA data structure

Name	Type	Description
I_Type	BOOL	Connected encoder type. TRUE: Asymetric encoder FALSE: Symetric encoder
I_PulseFactor	INT	Possible pulse factor for the encoder pulses. Values: 1, 2, and 4-fold evaluation.
I_IncPerTurn	INT	Increments per encoder rotation, (for encoder resolution, see encoder data sheet).
I_PathPerTurn	DINT	Actual path, which is covered by the machine per encoder rotation. This path should be specified in the smallest unit to be measured and must be less than I_IncPerTurn * I_PulseFactor.
I_Offset	DINT	Offset for the position values. This value can be used freely to shift the counting range of the module.

8.3 FUM_INC_DATA

The FUM_INC_DATA function block is used to assign the variables in the T_INC_IN_DATA data structure. This function block provides the exact variables of the data structure as inputs. The IN_INC_DATA input of this function block can be linked directly to the OUT_INC_DATA output of the INC_IN function block.

8.3.1 Function block call



8.3.2 T_INC_IN_DATA data structure

Name	Type	Description
O_Inp_1	BOOL	Status of digital input 1.
O_Inp_2	BOOL	Status of digital input 2.
O_Inp_3	BOOL	Status of digital input 3.
O_RefMarkFound	BOOL	The reference point has been found. The axis is homed.
O_IncValue	DINT	Direct position value of the terminal (not scaled).

9 Startup examples

For the startup instruction of the FunctionModule library please find the following examples:

- FUM_*_EXA_C_COUNT.pcwex
- FUM_*_EXA_C_FREQ.pcwex
- FUM_*_EXA_C_PULSE.pcwex
- FUM_*_EXA_C_TIME.pcwex
- FUM_*_EXA_IL_PWM2_PN.pcwex

These examples are packed in the zipped Examples folder of the library.

They describe the use of different function modules with function blocks of FunctionModule library.

9.1 Example 1: FUM_*_EXA_C_COUNT

9.1.1 Bus structure

For this example, the following hardware is used:

- AXC 3050 (2700989)
- AXL F DI8/1 DO8/1 1H (2701916)
- IL PN BK DI8 DO4 2TX-PAC (2703994)
- IB IL CNT-PAC (2861852)

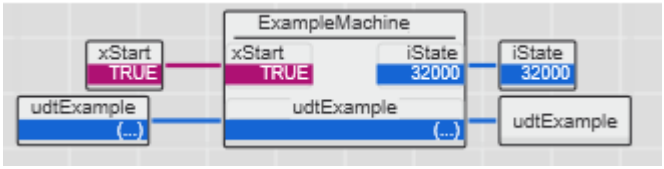
The wiring of the AXL F DI8/1 DO8/1 1H (2701916) terminal generates input signals that can be counted by the IB IL CNT-PAC (2861852) module with different operating modes.

9.1.2 Example description

The C_COUNT function block only supports “event counting” mode. In this example the IB IL CNT-PAC (2861852) terminal counts up between start value and end value.

9.1.2.1 Example machine

For starting ExampleMachine function block, xStart input has set to TRUE.



9.1.2.2 Example

```
CASE udtExample.iState OF
    0: (* Init *)
        IF xStart   = TRUE THEN
            (* Stimulus function block *)
            (* Cycle time *)
            udtExample.udtStim.tCycle       := TIME#120ms;
            (* Activation of function block *)
            udtExample.udtStim.xActivate    := TRUE;

            (* PC_Count function block *)
            (* Behavior in the event of an INTERBUS reset *)
            udtExample.udtFUM_PC_Count.xI_BehaviourBusReset     := FALSE;
            (* Possible input wiring *)
            udtExample.udtFUM_PC_Count.iI_InputSwitching        := 0;
            (* Possible input link *)
            udtExample.udtFUM_PC_Count.iI_SourceGateConnection  := 0;
            (* Gate input condition that must be met for the counting process *)
            udtExample.udtFUM_PC_Count.iI_Gate                  := 0;
            (* Single or continuous count *)
            udtExample.udtFUM_PC_Count.xI_Continuous            := FALSE;
            (* Counting direction: UP *)
            udtExample.udtFUM_PC_Count.xI_Direction             := TRUE;
            (* Switching behavior of the digital output *)
            udtExample.udtFUM_PC_Count.iI_Output                := 0;
            (* Possible times for the duration of the output puls *)
            udtExample.udtFUM_PC_Count.iI_PulseWidth            := 0;
```

```
            (* Start counting value *)
            udtExample.udtFUM_PC_Count.diI_StartValue           := DINT#0;
            (* Final counting value *)
            udtExample.udtFUM_PC_Count.diI_EndValue             := DINT#100;

            (* C_Count function block *)
            (* A positive edge restarts "event counting" mode *)
            udtExample.udtFUM_C_Count.xIN_START := TRUE;
            (* A positive edge stops "event counting" mode *)
            udtExample.udtFUM_C_Count.xIN_STOP  := FALSE;

            (* Activate counter function block *)
            udtExample.udtFUM_C_Count.xIN_ACTIVATE  := TRUE;
            (* Waiting for xReady *)
            IF udtExample.udtFUM_C_COUNT.xOUT_READY = TRUE THEN
                udtExample.iState   := 100;
            END_IF;
        END_IF;

    100: (* Checking for xOUT_Error *)
        IF udtExample.udtFUM_C_Count.xOUT_ERROR = TRUE THEN
            udtExample.iState   := 9000;
        ELSIF udtExample.udtFUM_C_COUNT.diOUT_VALUE =
            udtExample.udtFUM_PC_COUNT.diI_EndValue THEN
            udtExample.iState   := 32000;
        END_IF;

    9000: (* Error handling *)
        (* Insert your error handling here *)

        IF xStart   = FALSE THEN
            udtExample.iState   := 32000;
        END_IF;

    32000: (* Example is finished *)
        (* Deactivation of all function blocks *)
        udtExample.udtStim.xActivate                        := FALSE;
        udtExample.udtStim.tCycle                           := TIME#0ms;
        udtExample.udtFUM_PC_Count.xI_BehaviourBusReset     := FALSE;
        udtExample.udtFUM_PC_Count.iI_InputSwitching        := 0;
        udtExample.udtFUM_PC_Count.iI_SourceGateConnection  := 0;
        udtExample.udtFUM_PC_Count.iI_Gate                  := 0;
        udtExample.udtFUM_PC_Count.xI_Continuous            := FALSE;
        udtExample.udtFUM_PC_Count.xI_Direction             := FALSE;
        udtExample.udtFUM_PC_Count.iI_Output                := 0;
        udtExample.udtFUM_PC_Count.iI_PulseWidth            := 0;
        udtExample.udtFUM_PC_Count.diI_StartValue           := DINT#0;
        udtExample.udtFUM_PC_Count.diI_EndValue             := DINT#0;
        udtExample.udtFUM_C_Count.xIN_ACTIVATE              := FALSE;
        udtExample.udtFUM_C_Count.xIN_START                 := FALSE;
        udtExample.udtFUM_C_Count.xIN_STOP                  := FALSE;

        IF xStart   = FALSE THEN
            udtExample.iState   := 0;
        END_IF;
END_CASE;

iState  := udtExample.iState;
```

9.2 Example 2: FUM_*_EXA_C_FREQ

9.2.1 Bus structure

For this example, the following hardware is used:

- AXC 3050 (2700989)
- AXL F DI8/1 DO8/1 1H (2701916)
- IL PN BK DI8 DO4 2TX-PAC (2703994)
- IB IL CNT-PAC (2861852)

The wiring of the AXL F DI8/1 DO8/1 1H (2701916) terminal generates input signals that can be processed by the IB IL CNT-PAC (2861852) module with different operating modes.

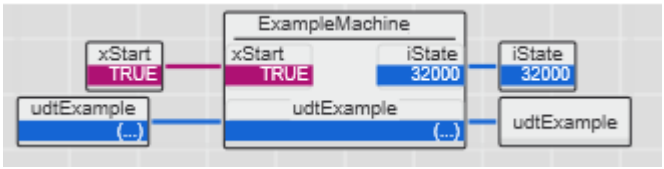
9.2.2 Example description

The C_FREQ function block only supports “frequency measurement” mode. In this example the IB IL CNT-PAC (2861852) terminal cyclically reads the measured values, which can be seen at OUT_VALUE output as frequency measured value.

The setting of IN_TIME_BASE input to 100 results in a gate time of 1s and a resolution of 1 Hz/LSB. The chosen tCycle time of 250ms results in a frequency of 1/500ms = 2 Hz.

9.2.2.1 Example machine

For starting ExampleMachine function block, xStart input has set to TRUE.



9.2.2.2 Example

```
CASE udtExample.iState OF
    0: (* Init *)
        IF xStart   = TRUE THEN
            (* Stimulus function block *)
            (* Cycle time *)
            udtExample.udtStim.tCycle       := TIME#250ms;
            (* Activation of function block *)
            udtExample.udtStim.xActivate    := TRUE;

            (* C_FREQ function block *)
            (* Behavior in the event of an INTERBUS reset *)
            udtExample.udtFUM_C_FREQ.xIN_BEHAVIOUR_BUS_RESET    := FALSE;
            (* Possible input wiring *)
            udtExample.udtFUM_C_FREQ.iIN_INPUT_SWITCHING        := INT#0;
            (* Possible input links *)
            udtExample.udtFUM_C_FREQ.iIN_SOURCE_GATE_CONNECTION := INT#0;
            (* Frequency measurement options *)
            udtExample.udtFUM_C_FREQ.iIN_TYPE                   := INT#1;
            (* Gate time for frequency measurement *)
            udtExample.udtFUM_C_FREQ.iIN_TIME_BASE              := INT#100;
            (* Activation of function blocks *)
            udtExample.udtFUM_C_FREQ.xIN_ACTIVATE               := TRUE;
```

```
            udtExample.iState   := 100;
        END_IF;

    100: (* Checking for xOUT_ERROR *)
        IF udtExample.udtFUM_C_FREQ.xOUT_ERROR  = TRUE THEN
            udtExample.iState   := 9000;
        ELSE
            udtExample.iState   := 32000;
        END_IF;

    9000: (* Error handling *)
        (* Insert your error handling here *)

        IF xStart   = FALSE THEN
            udtExample.iState   := 32000;
        END_IF;

    32000: (* Example is finished *)
        (* Deactivation of function blocks *)
        udtExample.udtStim.tCycle                           := TIME#0ms;
        udtExample.udtStim.xActivate                        := FALSE;
        udtExample.udtFUM_C_FREQ.xIN_ACTIVATE               := FALSE;
        udtExample.udtFUM_C_FREQ.xIN_BEHAVIOUR_BUS_RESET    := FALSE;
        udtExample.udtFUM_C_FREQ.iIN_INPUT_SWITCHING        := INT#0;
        udtExample.udtFUM_C_FREQ.iIN_SOURCE_GATE_CONNECTION := INT#0;
        udtExample.udtFUM_C_FREQ.iIN_TYPE                   := INT#0;
        udtExample.udtFUM_C_FREQ.iIN_TIME_BASE              := INT#0;

        IF xStart   = FALSE THEN
            udtExample.iState   := 0;
        END_IF;
END_CASE;

iState  := udtExample.iState;
```

9.3 Example 3: FUM_*_EXA_C_PULSE

9.3.1 Bus structure

For this example, the following hardware is used:

- AXC 3050 (2700989)
- IB IL CNT-PAC (2861852)

9.3.2 Example description

The C_PULSE function block only supports “pulse generator” mode. In this example the IB IL CNT-PAC (2861852) terminal generates a signal with the given PULSEuency at its output.

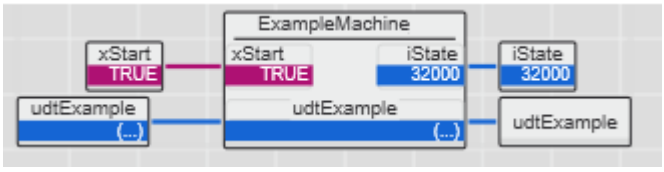
In this example the target PULSEuency at IN_PULSEUENCY input is specified with 2000 Hz.

The output value and operating mode can be seen at process data output OUT_OUTPUT_ADDRESS.

Please note, that in this example the PC Worx 6 project needs a cyclic task with an interval of 100 ms. Otherwise this resolution cannot be measured, as the task cycle time is too slow.

9.3.2.1 Example machine

For starting ExampleMachine function block, xStart input has set to TRUE.



9.3.2.2 Example

```
CASE udtExample.iState OF
    0: (* Init *)
        IF xStart   = TRUE THEN
            (* Behavior in the event of an INTERBUS reset *)
            udtExample.udtFUM_C_PULSE.xIN_BEHAVIOUR_BUS_RESET   := FALSE;
            (* Target frequency for "pulse generator" mode *)
            udtExample.udtFUM_C_PULSE.iIN_FREQUENCY             := INT#2000;
            (* Activation of function block *)
            udtExample.udtFUM_C_PULSE.xIN_ACTIVATE              := TRUE;

            udtExample.iState   := 100;
        END_IF;

    100: (* Checking for xOUT_ERROR *)
        IF udtExample.udtFUM_C_PULSE.xOUT_ERROR = TRUE THEN
            udtExample.iState   := 9000;
        ELSE
            udtExample.iState   := 32000;
        END_IF;

    9000: (* Error handling *)
        (* Insert your error handling here *)

        IF xStart   = FALSE THEN
```

```
            udtExample.iState   := 32000;
        END_IF;

    32000: (* Example is finished *)
        (* Deactivation of function blocks *)
        udtExample.udtFUM_C_PULSE.xIN_ACTIVATE              := FALSE;
        udtExample.udtFUM_C_PULSE.xIN_BEHAVIOUR_BUS_RESET   := FALSE;
        udtExample.udtFUM_C_PULSE.iIN_FREQUENCY             := INT#0;

        IF xStart   = FALSE THEN
            udtExample.iState   := 0;
        END_IF;
END_CASE;

iState  := udtExample.iState;
```

9.4 Example 4: FUM_*_EXA_C_TIME

9.4.1 Bus structure

For this example, the following hardware is used:

- AXC 3050 (2700989)
- AXL F DI8/1 DO8/1 1H (2701916)
- IL PN BK DI8 DO4 2TX-PAC (2703994)
- IB IL CNT-PAC (2861852)

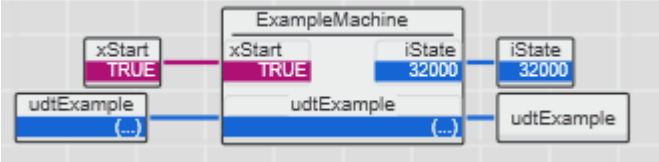
The wiring of the AXL F DI8/1 DO8/1 1H (2701916) terminal generates input signals that can be processed by the IB IL CNT-PAC (2861852) module with different operating modes.

9.4.2 Example description

The C_TIME function block only supports “time measurement” mode. The terminal measures the period or pulse time in a given resolution and timeline. Here the IB IL CNT-PAC (2861852) terminal offers the measured value of the parameterized operating mode period time measurement. Here, the iResolution input at PC_TIME function block has to be parameterized with a value of 10000.

9.4.2.1 Example machine

For starting ExampleMachine function block, xStart input has set to TRUE.



9.4.2.2 Example

```
CASE udtExample.iState OF
    0: (* Init *)
        IF xStart   = TRUE THEN
            (* Stimulus function block *)
            (* Cycle time *)
            udtExample.udtStim.tCycle   := TIME#300ms;

            (* PC_TIME function block *)
            (* Behavior in the event of an INTERBUS reset *)
            udtExample.udtFUM_PC_TIME.xI_BehaviourBusReset      := FALSE;
            (* Possible input wiring *)
            udtExample.udtFUM_PC_TIME.iI_InputSwitching         := INT#0;
            (* Possible input links *)
            udtExample.udtFUM_PC_TIME.iI_SourceGateConnection   := INT#0;
            (* Resolution value of the LSB *)
            udtExample.udtFUM_PC_TIME.iI_Resolution             := INT#10000;
            (* Switching behavior of the digital output *)
            udtExample.udtFUM_PC_TIME.xI_BehaviourOutput        := FALSE;
            (* Type of measurement *)
            udtExample.udtFUM_PC_TIME.xI_MeasurementType        := FALSE;
            (* Display behavior of limit values *)
            udtExample.udtFUM_PC_TIME.iI_Mismatch               := INT#0;
            (* Standardized start value *)
            udtExample.udtFUM PC TIME.diI StartValue            := DINT#0;
```

```
            (* Standardized final value *)
            udtExample.udtFUM_PC_TIME.diI_EndValue              := DINT#0;

            (* C_TIME function block *)
            (* Setting this input resets the mismatch message *)
            udtExample.udtFUM_C_TIME.xIN_MISMATCH_RESET := FALSE;
            (* Activation of function block *)
            udtExample.udtFUM_C_TIME.xIN_ACTIVATE       := TRUE;

            udtExample.iState   := 100;
        END_IF;

    100: (* Checking for error *)
        IF udtExample.udtFUM_C_TIME.xOUT_ERROR  = TRUE THEN
            (* Go to error state *)
            udtExample.iState   := 9000;
        ELSE
            (* Activation of function block *)
            udtExample.udtStim.xActivate    := TRUE;
            (* Go to finish *)
            udtExample.iState   := 32000;
        END_IF;

    9000: (* Error handling *)
        (* Insert your error handling here *)

        IF xStart   = FALSE THEN
            udtExample.iState   := 32000;
        END_IF;

    32000: (* Example is finished *)
        (* Deactivation of function blocks *)
        udtExample.udtStim.tCycle                           := TIME#0ms;
        udtExample.udtFUM_PC_TIME.xI_BehaviourBusReset      := FALSE;
        udtExample.udtFUM_PC_TIME.iI_InputSwitching         := INT#0;
        udtExample.udtFUM_PC_TIME.iI_SourceGateConnection   := INT#0;
        udtExample.udtFUM_PC_TIME.iI_Resolution             := INT#0;
        udtExample.udtFUM_PC_TIME.xI_BehaviourOutput        := FALSE;
        udtExample.udtFUM_PC_TIME.xI_MeasurementType        := FALSE;
        udtExample.udtFUM_PC_TIME.iI_Mismatch               := INT#0;
        udtExample.udtFUM_PC_TIME.diI_StartValue            := DINT#0;
        udtExample.udtFUM_PC_TIME.diI_EndValue              := DINT#0;
        udtExample.udtFUM_C_TIME.xIN_MISMATCH_RESET         := FALSE;
        udtExample.udtFUM_C_TIME.xIN_ACTIVATE               := FALSE;

        IF xStart   = FALSE THEN
            udtExample.iState   := 0;
        END_IF;
END_CASE;

iState  := udtExample.iState;
```

9.5 Example 5: FUM_*_EXA_IL_PWM2_PN

9.5.1 Bus structure

For this example, the following hardware is used:

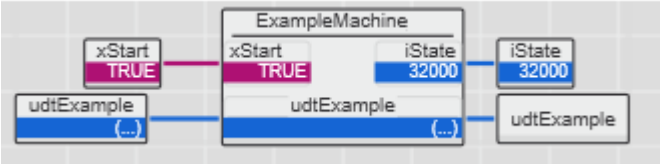
- AXC 3050 (2700989)
- IL PN BK DI8 DO4 2TX-PAC (2703994)
- IB IL PWM/2-PAC (2861632)

9.5.2 Example description

In this example, the IB IL PWM2 function block should be operated in two different operating modes. Channel 1 is set to PWM operating mode, whereby a square-wave signal should be output which is 500 ms TRUE and 500 ms FALSE. Channel 2 is set to Single Shot operating mode. The output here should be set to TRUE for five seconds if the xPulseStart input is TRUE.

9.5.2.1 Example machine

For starting ExampleMachine function block, xStart input has set to TRUE.



9.5.2.2 Example

```
CASE udtExample.iState OF
    0: (* Init *)
        IF xStart   = TRUE THEN
            (* FUM_IL_PWM_Para function blocks *)
            FOR iCH := 1 TO 2 DO
                (* Frequency mode *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].iFrequency          := INT#0;
                (* Pulse direction mode: adjustment of rotation direction *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].xDirection          := FALSE;
                (* Pulse direction mode: reset of positioning counter *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].xResetPosition      := FALSE;
                (* Pulse direction mode: activates the output with the
                set control frequency *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].xStartMovement      := FALSE;
                (* Pulse direction mode: control frequency *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].iPulseFrequency     := INT#0;
                (* Pulse direction mode: target position of the motor *)
                udtExample.arrFUM_IL_PWM2_Para[iCH].uiTargetPosition    := UINT#0;
            END_FOR;

            (* CH1 *)
            (* Channel selection for parameterization *)
            udtExample.arrFUM_IL_PWM2_Para[1].iChannel      := INT#1;
            (* Terminal block operating mode for one channel *)
            udtExample.arrFUM_IL_PWM2_Para[1].iMode         := INT#1;
            (* PWM mode: values for the frequency specification *)
            udtExample.arrFUM_IL_PWM2_Para[1].iCycleTime    := INT#20;
            (* PWM mode: channel pulse ratio *)
```

```
        (* Single shot mode: time scale for pulse length *)
        udtExample.arrFUM_IL_PWM2_Para[1].iTimeBase     := INT#0;
        (* Single shot mode: pulse length depending on iTimeBase *)
        udtExample.arrFUM_IL_PWM2_Para[1].iTimeFactor   := INT#0;
        (* Single shot mode: starting pulse output *)
        udtExample.arrFUM_IL_PWM2_Para[1].xPulseStart   := FALSE;

        (* CH2 *)
        (* Channel selection for parameterization *)
        udtExample.arrFUM_IL_PWM2_Para[2].iChannel      := INT#2;
        (* Terminal block operating mode for one channel *)
        udtExample.arrFUM_IL_PWM2_Para[2].iMode         := INT#3;
        (* PWM mode: values for the frequency specification *)
        udtExample.arrFUM_IL_PWM2_Para[2].iCycleTime    := INT#0;
        (* PWM mode: channel pulse ratio *)
        udtExample.arrFUM_IL_PWM2_Para[2].iPulseRatio   := INT#0;
        (* Single shot mode: time scale for pulse length *)
        udtExample.arrFUM_IL_PWM2_Para[2].iTimeBase     := INT#4;
        (* Single shot mode: pulse length depending on iTimeBase *)
        udtExample.arrFUM_IL_PWM2_Para[2].iTimeFactor   := INT#50;
        (* Single shot mode: starting pulse output *)
        udtExample.arrFUM_IL_PWM2_Para[2].xPulseStart   := TRUE;

        (* FUM_IL_PWM_2 function block *)
        (* Activation of function block *)
        udtExample.udtFUM_IL_PWM2.xActivate := TRUE;
        (* Reset of function block *)
        udtExample.udtFUM_IL_PWM2.xReset    := FALSE;
        (* Channel selection *)
        udtExample.udtFUM_IL_PWM2.iChSelect := INT#0;

        udtExample.iState   := udtExample.iState;
    END_IF;

100: (* Checking for xError *)
    IF udtExample.udtFUM_IL_PWM2.xError = TRUE THEN
        udtExample.iState   := 9000;
    ELSE
        udtExample.iState   := 32000;
    END_IF;

9000: (* Error handling *)
    (* Insert your error handling here *)

    IF xStart   = FALSE THEN
        udtExample.iState   := 32000;
    END_IF;

32000: (* Example is finished *)
    (* Deactivation of function blocks *)
    FOR iCH := 1 TO 2 DO
        udtExample.arrFUM_IL_PWM2_Para[iCH].iFrequency          := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].xDirection          := FALSE;
        udtExample.arrFUM_IL_PWM2_Para[iCH].xResetPosition      := FALSE;
        udtExample.arrFUM_IL_PWM2_Para[iCH].xStartMovement      := FALSE;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iPulseFrequency     := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].uiTargetPosition    := UINT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iChannel            := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iMode               := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iCycleTime          := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iPulseRatio         := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iTimeBase           := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].iTimeFactor         := INT#0;
        udtExample.arrFUM_IL_PWM2_Para[iCH].xPulseStart         := FALSE;
```

```
        END_FOR;

        udtExample.udtFUM_IL_PWM2.xActivate := FALSE;
        udtExample.udtFUM_IL_PWM2.xReset    := FALSE;
        udtExample.udtFUM_IL_PWM2.iChSelect := INT#0;

        IF xStart   = FALSE THEN
            udtExample.iState   := 0;
        END_IF;
END_CASE;

iState  := udtExample.iState;
```

10 Appendix

10.1 Data types

```
TYPE

FM_arr_Bit_0_15 : ARRAY[0..15] OF BOOL;
FUM_ARR_W_1_2   : ARRAY[1..2] OF WORD;
FUM_ARR_X_1_2   : ARRAY[1..2] OF BOOL;
FUM_ARR_I_1_2   : ARRAY[1..2] OF INT;

(*
** FUM_INC_IN
*)

T_INC_IN_PARA : STRUCT
    I_Type          : BOOL; (*Connected encoder type*)
    I_PulseFactor   : INT;  (*Possible pulse factor*)
    I_IncPerTurn    : INT;  (*Increments per encoder rotation*)
    I_PathPerTurn   : DINT; (*Actual path, which is covered*)
    I_Offset        : DINT; (*Offset for the position values*)
END_STRUCT;

T_INC_IN_DATA : STRUCT
    O_Inp_1         : BOOL; (*Status of digital input 1*)
    O_Inp_2         : BOOL; (*Status of digital input 2*)
    O_Inp_3         : BOOL; (*Status of digital input 3*)
    O_RefMarkFound  : BOOL; (*The reference point has been found*)
    O_IncValue      : DINT; (*Direct position value of the terminal*)
END_STRUCT;

(*
** FUM_C_TIME
*)

STR_C_TIME : STRUCT
    I_BehaviourBusReset     : BOOL; (* Behavior of the bus reset *)
    I_InputSwitching        : INT;  (* Possible input selection *)
    I_SourceGateConnection  : INT;  (* Possible input connection *)
    I_Resolution            : INT;  (* Resolution of the LSB *)
    I_BehaviourOutput       : BOOL; (* Bahavior from the digital output *)
    I_MeasurementType       : BOOL; (* Type of measurement *)
    I_Mismatch              : INT;  (* Behavior of the mismatch function *)
    I_StartValue            : DINT; (* Start Value for mismatch function *)
    I_EndValue              : DINT; (* End Value for mismatch function *)
END_STRUCT;

(*
** FUM_P_TIME
*)

STR_P_TIME : STRUCT
    I_BehaviourBusReset     : BOOL; (* Behavior of the bus reset *)
    I_InputSwitching        : INT;  (* Possible input selection *)
    I_SourceGateConnection  : INT;  (* Possible input connection *)
    I_Resolution            : INT;  (* Resolution of the LSB *)
    I_BehaviourOutput       : BOOL; (* Bahavior from the digital output *)
    I_MeasurementType       : BOOL; (* Type of measurement *)
    I_Mismatch              : INT;  (* Behavior of the mismatch function *)
    I_StartValue            : DINT; (* Start Value for mismatch function *)
    I_EndValue              : DINT; (* End Value for mismatch function *)
END_STRUCT;
```

```
(*
** FUM_C_COUNT
*)

STR_C_COUNT : STRUCT
    I_BehaviourBusReset     : BOOL; (* Behavior of the bus reset *)
    I_InputSwitching        : INT;  (* Possible input selection *)
    I_SourceGateConnection  : INT;  (* Possible input connection *)
    I_Gate                  : INT;  (* Condition for the gate input *)
    I_Continuous            : BOOL; (* Counter continuous mode *)
    I_Direction             : BOOL; (* Counter direction *)
    I_Output                : INT;  (* Behavior from the digital output *)
    I_PulseWidth            : INT;  (* Possible time for the digital output *)
    I_StartValue            : DINT; (* Start Value for counter (24 Bit) *)
    I_EndValue              : DINT; (* End Value for counter (24 Bit) *)
END_STRUCT;

(*
** FUM_IL_DI_8_S0
*)

S0_arr_UD_0_8 : ARRAY[0..7] OF UDINT;

(* Extended information about each channel configuration *)

Extended_Config_Info : STRUCT
    OperatingCounter        : BOOL; (* Op Mode = Operating Counter *)
    PeriodLengthMeasuring   : BOOL;
        (* Op Mode = Pulse Counter with period length measurement *)
    PulseLengthMeasuring    : BOOL;
        (* Op Mode = Pulse Counter with pulse length measurement *)
    ChannelInvert           : BOOL; (* Inversion of inputs *)
    Debouncing              : INT;  (* Debouncing of inputs *)
END_STRUCT;

S0_arr_RD_CFG : ARRAY[0..7] OF Extended_Config_Info ;

udtChannel : STRUCT
    udiCounter  : UDINT;
    uiDuration  : UINT;
    xPreSet     : BOOL;
    xINV        : BOOL;
    xON         : BOOL;
    xGC         : BOOL;
    bDEB        : BYTE;
    udiPreSet   : UDINT;
END_STRUCT;

arrDataDI_8_S0 : ARRAY[0..7] OF udtChannel;

arrTest : ARRAY[0..7] OF DWORD;

(*
** FUM_PWM
*)

FUM_UDT_PWM2_PARA : STRUCT      (* structure for channel parameters *)
    iMode               : INT;  (* op mode *)
    iCycleTime          : INT;  (* cycle time, op mode 1 PWM *)
    iPulseRatio         : INT;  (* pulse ratio, op mode 1 PWM *)
    iFrequency          : INT;  (* frequency, op mode 2 frequency *)
    iTimeBase           : INT;  (* time base, op mode 3 single shot *)
    iTimeFactor         : INT;  (* time factor, op mode 3 single shot *)
```

```
    xPulseStart         : BOOL; (* pulse start, op mode 3 single shot *)
    xDirection          : BOOL; (* direction, op mode 4 pulse direction *)
    xResetPosition      : BOOL; (* reset position, op mode 4 pulse direction *)
    xStartMovement      : BOOL; (* start movement, op mode 4 pulse direction *)
    iPulseFrequency     : INT;  (* pulse frequency, op mode 4 pulse direction *)
    uiTargetPosition    : UINT; (* target position, op mode 4 pulse direction *)
END_STRUCT;

FUM_UDT_PWM2_DIAG : STRUCT  (* structure for channel diagnosis *)
    xActive         : BOOL; (* selected channel is active *)
    xPulseState     : BOOL; (* status selected channel pulse output *)
    xError          : BOOL; (* slected channel indicate an error *)
    wDiagCode       : WORD; (* selected channel diagnosis code *)
    wAddDiagCode    : WORD; (* selected channel additional diagnosis code *)
END_STRUCT;

FUM_ARR_PWM2_PARA_1_2   : ARRAY[1..2] OF FUM_UDT_PWM2_PARA;
FUM_ARR_PWM2_DIAG_1_2   : ARRAY[1..2] OF FUM_UDT_PWM2_DIAG;

FUM_UDT_PWM2_DATA : STRUCT (* main structure *)
    arrPWM2_PARA    : FUM_ARR_PWM2_PARA_1_2;
    arrPWM2_DIAG    : FUM_ARR_PWM2_DIAG_1_2;
    wDiagCode       : WORD;                 (* global diagnosis code *)
    wAddDiagCode    : WORD;                 (* global extended diagnosis code *)
END_STRUCT;

END_TYPE
```

11 Support

For technical support please contact your local PHOENIX CONTACT agency

at https://www.phoenixcontact.com

Owner:

PHOENIX CONTACT Electronics GmbH
Business Unit Automation Systems
System Services
Library Services

In case of a support request we need:

- Development system with
  - Name (e.g. PC Worx, PLCnext Engineer)
  - Version (e.g. PLCnext Engineer 2022.0.1 LTS)
- Bus structure / plant including all articles with
  - Name
  - Order number
  - Firmware version
- External components
- Used libraries with
  - Name
  - Version (e.g. IOL_Basic_7)
- Detailed problem description with
  - Diag codes of all function blocks in error state