Operating Instructions

SICK
AFS60/AFM60 EtherNet/IP
Add-On Instructions

SICK_AFX60 Add-On Instructions for
Rockwell Automation controllers

Edition: July 2012

SICK
Sensor Intelligence.

Table of contents

1 About This Document

Please read this chapter carefully before you begin working with these operating instructions and the SICK_AFX60 Add-On Instructions.

1.1 Function of This Document

These operating instructions describe how to use the SICK_AFX60 Add-On Instructions. They are used to guide technical personnel working for the machine manufacturer/operator in project planning and commissioning the Add-On Instructions.

1.2 Target Group

These operating instructions are aimed at specialist personnel such as technicians and engineers.

2 General Information

The Add-On Instructions (AOI) enable communication between a Rockwell controller and the SICK AFS60/AFM60 EtherNet/IP encoder. In order to do so, the encoder must be integrated into the control project's EtherNet/IP environment. The communication is realized with the encoder non-cyclically via the CIP protocol.

The AOI enables selected sensor data to be read-out from the device and described as required.

This module can be used to read out the following device parameters:
- Serial number
- Flag scaling
- Number of steps per revolution
- Total resolution
- Preset value
- Lower position limit
- Upper position limit
- Speed format
- Speed lower limit value
- Speed upper limit value
- Warnings
- Warning flag
- Encoder temperature
- Encoder temperature format
- Encoder movement time
- Encoder operating time
- Highest speed since commissioning

This module can be used to write the following device parameters:
- Preset value
- Lower position limit
- Upper position limit
- Speed lower limit value
- Speed upper limit value
- Encoder temperature format

2.1 Module Specifications

AOI Name:	SICK_AFX60
Version:	1.0
Routine name:	Logic
UDTs:	SICK_AFX60_DATA
	SICK_AFX60_READ_SEL
	SICK_AFX60_READ
	SICK_AFX60_WRITE_SEL
	SICK_AFX60_WRITE
Module call:	cyclical
Language:	Structured Text (ST)

The AFX60 AOI is an asynchronously operating routine, i.e. processing is spread over numerous PLC cycles. The routine must be called by the user program until processing is completed.

The following figure shows how the AOI is represented in the relay ladder logic.

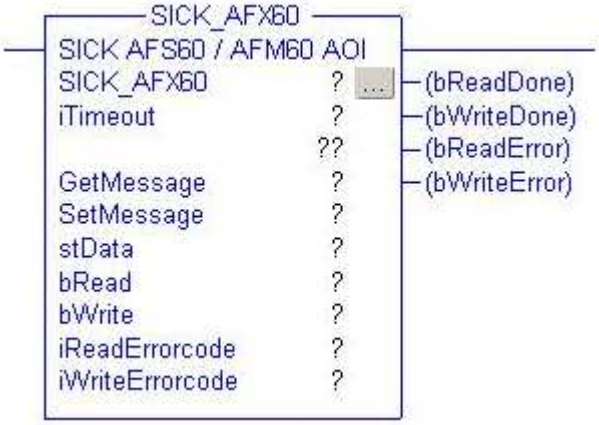

Figure 1: Representation of SICK_AFX60 AOI

3 Integration of AOI in RSLogix5000

The AFS60/AFS60 encoder implementation is managed via the Add-On Instructions (AOI). This AOI includes a program routine and numerous data structures (UDTs). The program routine can be called from any part of the user program and enables non-cyclical data exchange with the sensor (CIP generic).

3.1 Hardware Configuration

Please refer to the AFS60/AFM60 encoder operating manual for information about the hardware configuration. The encoder's cyclical process data is available as soon as the sensor has been integrated into the LSLogix5000. The AOI additionally supports access to the encoder's non-cyclical service data.

3.2 AOI Import

In order to enable use of the SICK_AFX60 AOI in the user program, it is first necessary to import "File → Import Component → Add-On Instruction" into an existing project.

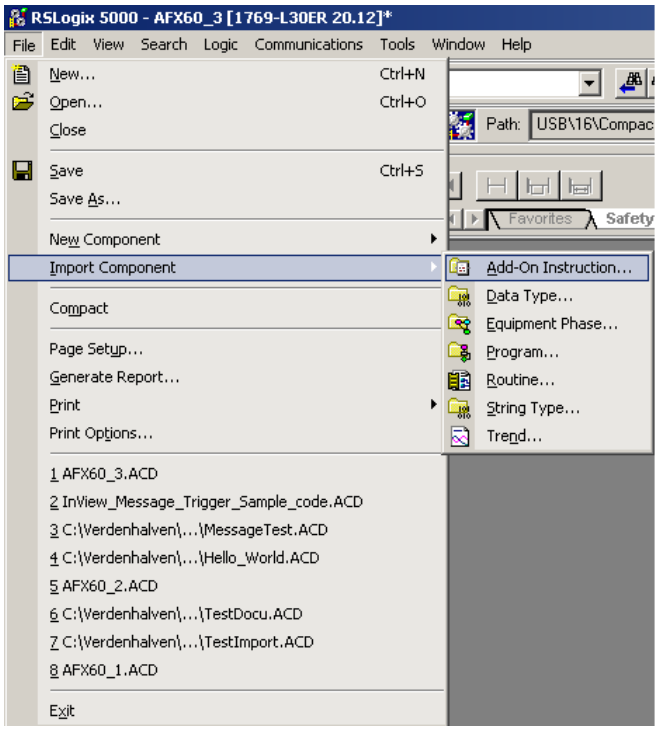

Figure 2: Import the SICK_AFX60 Add-On Instructions

3.3 Calling the module

In order to utilize the SICK_AFX60 AOI, it must first be called in the user program and activated with valid parameters. It is possible to read out and describe sensor parameters after valid activation has been completed.

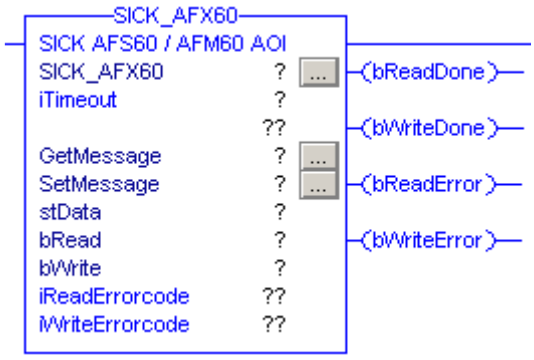

Figure 3: View of the SICK_AFX60 routine in the user program

SICK_AFX60: Name of module instance. The module instance enables access to the routine's input and output parameters.

iTimeout: Time after a timeout error was triggered, if the processing time exceeds this time. The timeout value is expressed in milliseconds (normal value: 5000ms).

GetMessage: Additionally to activating the parameter, it is also necessary to configure it via the button on the right marked "…".

SetMessage: Additionally to activating the parameter, it is also necessary to configure it via the button on the right marked "…".

stData: The "stData" parameter uses the structure of the supplied SICK_AFX_DATA UDTs. The parameter values that have been read-out and are pending writing to the encoder are stored here. The UDT must be instanced as Controller Tag and be transferred to the routine.

bRead: All of the sensor parameters supported by the AOI are read-out via a rising edge.

bWrite: All of the selected parametric values are written to the encoder via a rising edge in data type "stData".

Figure 4 shows a sample circuit for the routine in the user program (ladder logic).

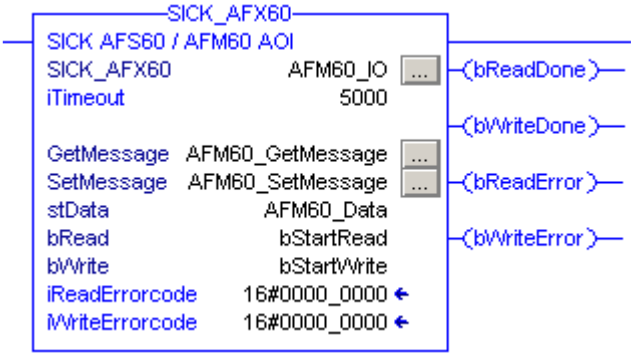

Figure 4: Sample circuit of SICK_AFX60 routine

3.3.1 Configuration of GetMessage Parameter

The GetMessage parameter must be configured once as follows via the "…" button (in addition to assignment).

Message Type:	CIP Generic
Service Type:	Get Attribute Single
Service Code:	Is automatically preset by the service types to 16#E
Instance:	1
Class:	1
Attribute:	1
Destination Element:	Reference to array "arrReadRecord" from the data structure which is transferred to input parameter "stData".
Path:	Hardware name of the corresponding encoder assigned in the RSLogix5000. The encoder can be selected via the "Browse…" button.
Connected:	Not active: The connection will be connected and disconnected automatically on request. Active: The connection will be maintained permanently (processing time for establishing and dropping the connection is irrelevant).

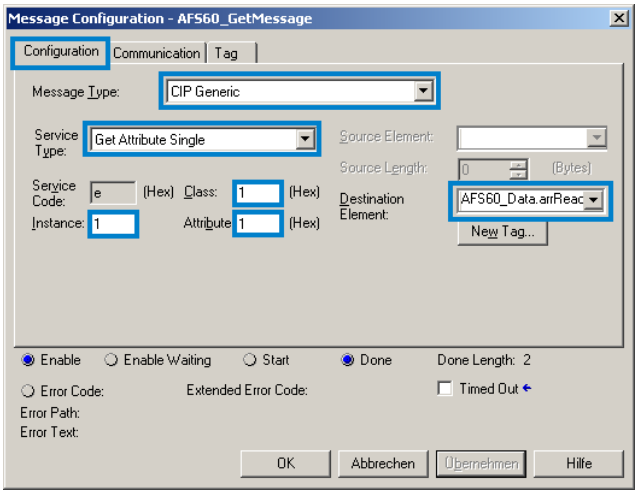

Figure 5: Configuration of GetMessage parameter (configuration)

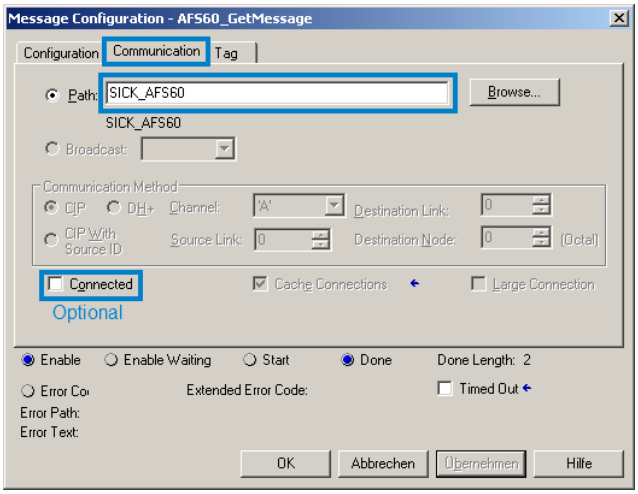

Figure 6: Configuration of GetMessage parameter (communication)

3.3.2 Configuration of SetMessage Parameter

The SetMessage parameter must be configured once as follows via the "…" button (in addition to assignment).

Message Type:	CIP Generic
Service Type:	Set Attribute Single
Service Code:	Is automatically preset by the service types to 16#10
Instance:	1
Class:	1
Attribute:	1
Source Element:	Reference to array "arrWriteRecord" from the data structure which is transferred to input parameter "stData".
Source Length:	4
Path:	Hardware name of the corresponding encoder assigned in the RSLogix5000. The encoder can be selected via the "Browse…" button.
Connected:	Not active: The connection will be connected and disconnected automatically on request. Active: The connection will be maintained permanently (processing time for establishing and dropping the connection is irrelevant).

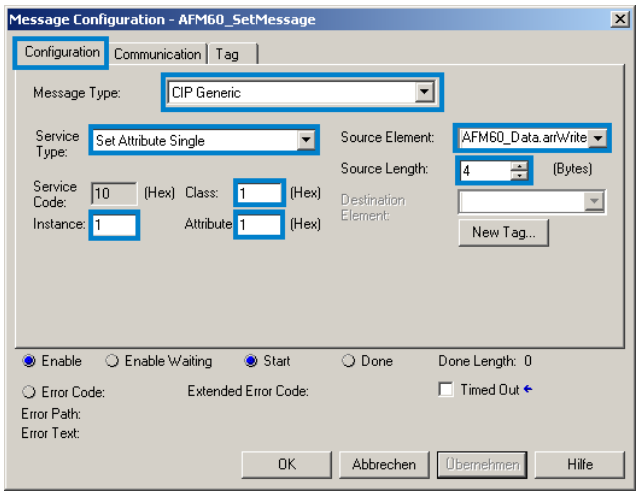

Figure 7: Configuration of SetMessage parameter (configuration)

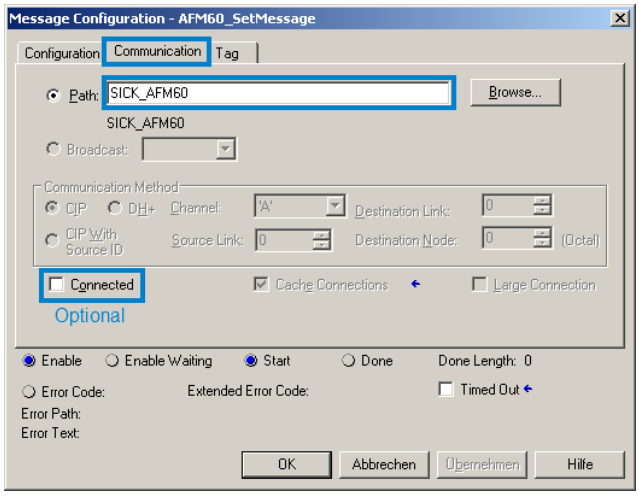

Figure 8: Configuration of SetMessage parameter (communication)

3.4 Reading Device Parameters

Before the module command "Read parameter" can be executed, it is first necessary to specify which device parameters should be read-out. This selection is made in the assigned data structure "stData" in the sub-structure "GetData.Selection".

If a parameter is assigned the value "1" in sub-structure "GetData.Selection", this means that the selected parameter will be read out when the module command is executed. If the bit "ReadAll" is set, then all of the device parameters listed in the "GetData" sub-structure are preselected for reading out.

Figure 9 shows a sample procedure for reading out the device parameters "PositionLowLimit", "PositionHighLimit" and "TemperatureValue".

Name	Value	Style	Data Type	Description
AFS60_Data	{...}		SICK_AFX60_DATA	AFS60 / AFM60 data structure
AFS60_Data.GetData	{...}		SICK_AFX60_READ	AFS60 / AFM60 data structure Reading sensor parameter v
AFS60_Data.GetData.Selection	{...}		SICK_AFX60_READ_SEL	AFS60 / AFM60 data structure Selection of the parameters t
AFS60_Data.GetData.Selection.ReadAll	0	Decimal	BOOL	AFS60 / AFM60 data structure Read all supported paramete
AFS60_Data.GetData.Selection.SerialNumber	0	Decimal	BOOL	AFS60 / AFM60 data structure Serial number (ID 6)
AFS60_Data.GetData.Selection.ScalingFunction	0	Decimal	BOOL	AFS60 / AFM60 data structure Scaling (14)
AFS60_Data.GetData.Selection.CountsPerRange	0	Decimal	BOOL	AFS60 / AFM60 data structure Number of steps per revolutio
AFS60_Data.GetData.Selection.TotalMeasuringRange	0	Decimal	BOOL	AFS60 / AFM60 data structure Total resolution (ID 17)
AFS60_Data.GetData.Selection.PresetValue	0	Decimal	BOOL	AFS60 / AFM60 data structure Preset value (ID 19)
AFS60_Data.GetData.Selection.PositionLowLimit	1	Decimal	BOOL	AFS60 / AFM60 data structure Lower limit for the position (ID
AFS60_Data.GetData.Selection.PositionHighLimit	1	Decimal	BOOL	AFS60 / AFM60 data structure Upper limit for the position (ID
AFS60_Data.GetData.Selection.VelocityFormat	0	Decimal	BOOL	AFS60 / AFM60 data structure Velocity unit (ID 25)
AFS60_Data.GetData.Selection.MinVelocitySetpoint	0	Decimal	BOOL	
AFS60_Data.GetData.Selection.MaxVelocitySetpoint	0	Decimal	BOOL	
AFS60_Data.GetData.Selection.Warnings	0	Decimal	BOOL	
AFS60_Data.GetData.Selection.WarningFlag	0	Decimal	BOOL	AFS60 / AFM60 data structure Warning flag (ID 49)
AFS60_Data.GetData.Selection.TemperatureValue	1	Decimal	BOOL	AFS60 / AFM60 data structure Actual temperature (ID 100)
AFS60_Data.GetData.Selection.TemperatureValueFor...	0	Decimal	BOOL	AFS60 / AFM60 data structure Temperature format (ID 101)
AFS60_Data.GetData.Selection.MotionTime	0	Decimal	BOOL	AFS60 / AFM60 data structure Saved motion time (ID 107)
AFS60_Data.GetData.Selection.OperatingTime	0	Decimal	BOOL	AFS60 / AFM60 data structure Saved operating time (ID 10
AFS60_Data.GetData.Selection.MaxVelocity	0	Decimal	BOOL	AFS60 / AFM60 data structure Highest velocity that the enc
AFS60_Data.GetData.SerialNumber	{...}	Decimal	SINT[4]	AFS60 / AFM60 data structure Serial number in the format [\
AFS60_Data.GetData.ScalingFunction	0	Decimal	BOOL	AFS60 / AFM60 data structure Scaling: 0 = Off; 1 = On
AFS60_Data.GetData.CountsPerRange	0	Decimal	DINT	AFS60 / AFM60 data structure Number of steps per revolutio
AFS60_Data.GetData.TotalMeasuringRange	0	Decimal	DINT	AFS60 / AFM60 data structure Total resolution
AFS60_Data.GetData.PresetValue	0	Decimal	DINT	AFS60 / AFM60 data structure Preset value: [0 ... Total mea
AFS60_Data.GetData.PositionLowLimit	0	Decimal	DINT	AFS60 / AFM60 data structure Lower limit for the position
AFS60_Data.GetData.PositionHighLimit	500	Decimal	DINT	AFS60 / AFM60 data structure Upper limit for the position
AFS60_Data.GetData.VelocityFormat	16#0000	Hex	INT	AFS60 / AFM60 data structure Velocity unit: 1F04h = counts
AFS60_Data.GetData.MinVelocitySetpoint	0	Decimal	DINT	
AFS60_Data.GetData.MaxVelocitySetpoint	0	Decimal	DINT	
AFS60_Data.GetData.Warnings	0	Decimal	INT	
AFS60_Data.GetData.WarningFlag	0	Decimal	BOOL	AFS60 / AFM60 data structure Warnings: 0 = No warning; 1
AFS60_Data.GetData.TemperatureValue	4200	Decimal	INT	AFS60 / AFM60 data structure Actual temperature with +-5%
AFS60_Data.GetData.TemperatureValueFormat	16#0000	Hex	INT	AFS60 / AFM60 data structure Temperature format: 1200h =
AFS60_Data.GetData.MotionTime	0	Decimal	DINT	AFS60 / AFM60 data structure Saved motion time in seconc
AFS60_Data.GetData.OperatingTime	0	Decimal	DINT	AFS60 / AFM60 data structure Saved operating time in secc
AFS60_Data.GetData.MaxVelocity	0	Decimal	DINT	AFS60 / AFM60 data structure Highest velocity that the enc
AFS60_Data.SetData	{...}		SICK_AFX60_WRITE	AFS60 / AFM60 data structure Parameter values that should
AFS60_Data.arrReadRecord	{...}	Decimal	SINT[4]	AFS60 / AFM60 data structure Read record for the GetMess
AFS60_Data.arrWriteRecord	{...}	Decimal	SINT[4]	AFS60 / AFM60 data structure Write record for the SetMess

Preselection of the parameters to be read

Reading result after triggering the AOI on the bRead bit

Figure 9: Reading out device parameters

For each preselection in sub-structure "GetData.Selection" there is a corresponding value in sub-structure "GetData", where the read-out values are stored.

The input parameter bRead must be triggered with a rising edge (signal change from logic zero to one) in order to execute the module command. The module signals the bReadDone = TRUE output parameter when the function has been successfully completed. The read results are stored in the transferred TAG in input "stData" (sub-structure "GetData"). A description of the device parameters can be found in the corresponding device documentation.

3.5 Writing Device Parameters

Before the module command "Write parameter" can be executed, it is first necessary to specify which device parameters should be written to. This selection is made in the assigned data structure "stData" in the sub-structure "SetData.Selection".

If a parameter is assigned the value "1" in sub-structure "SetData.Selection", this means that the selected parameter will be written to when the module command is executed. One or more parameters may be preselected.

For each preselection in sub-structure "SetData.Selection" there is a corresponding value in sub-structure "SetData". A description of the device parameters can be found in the corresponding device documentation.

Figure 10 shows a sample procedure for writing the device parameters "PresetValue" and "PositionHighLimit".

Name	Value	Style	Data Type	Description
AFS60_Data	{...}		SICK_AFX60_DATA	AFS60 / AFM60 data structure
AFS60_Data.GetData	{...}		Preselection of the parameters to be written	
AFS60_Data.SetData	{...}			
AFS60_Data.SetData.Selection	{...}		SICK_AFX60_WRITE_SEL	AFS60 / AFM60 data structure Selection of the
AFS60_Data.SetData.Selection.PresetValue	1	Decimal	BOOL	AFS60 / AFM60 data structure Preset value (ID
AFS60_Data.SetData.Selection.PositionLowLimit	0	Decimal	BOOL	AFS60 / AFM60 data structure Lower limit for th
AFS60_Data.SetData.Selection.PositionHighLimit	1	Decimal	BOOL	AFS60 / AFM60 data structure Upper limit for th
AFS60_Data.SetData.Selection.MinVelocitySetpoint	0	Decimal	BOOL	AFS60 / AFM60 data structure Minimum velocit
AFS60_Data.SetData.Selection.MaxVelocitySetpoint	0	Decimal	BOOL	AFS60 / AFM60 data structure Maximum veloci
AFS60_Data.SetData.Selection.TemperatureValueFor...	0	Decimal	BOOL	AFS60 / AFM60 data structure Temperature for
AFS60_Data.SetData.PresetValue	123	Decimal	DINT	AFS60 / AFM60 data structure Preset value: [0
AFS60_Data.SetData.PositionLowLimit	0	Decimal	DINT	AFS60 / AFM60 data structure Lower limit for th
AFS60_Data.SetData.PositionHighLimit	800	Decimal	DINT	AFS60 / AFM60 data structure Upper limit for th
AFS60_Data.SetData.MinVelocitySetpoint	0	Decimal	DINT	AFS60 / AFM60 data structure Minimum velocit
AFS60_Data.SetData.MaxVelocitySetpoint	0	Decimal	Values to be written	
AFS60_Data.SetData.TemperatureValueFormat	16#0000	Hex		
AFS60_Data.arrReadRecord	{...}	Decimal	SINT[4]	AFS60 / AFM60 data structure Read record for
AFS60_Data.arrWriteRecord	{...}	Decimal	SINT[4]	AFS60 / AFM60 data structure Write record for

Figure 10: Writing device parameters

The bWrite input parameter must be triggered with a rising edge (signal change from logic null to one) in order to carry out the write parameter procedure. The output parameter bWriteDone = TRUE signalizes that the command had been successfully executed.

3.6 Response to Errors

If the program routine has an incorrect input value or faulty input circuit, an error bit (bReadError, bWriteError) is set and an error code (iReadErrorcode, iWriteErrorcode) is given. No further processing is carried out. The diagnostic parameters (bReadError, bWriteError, iReadErrorcode, iWriteErrorcode) for the routine are retained until a new command is started.

4 Overview of Parameters

Parameter	Declaration	Data Type	Description
Enable	Input	BOOL	Enable Input (only for ladder logic)
iTimeout	Input	DINT	Time after a timeout error was triggered, if the module processing time exceeds this time. The timeout value is expressed in milliseconds. Normal value: 5000 [ms]
GetMessage	InOut	MESSAGE	Instance for the internally utilized MSG command for reading out a sensor parameter. This parameter must be configured as described in chapter 3.3.1.
SetMessage	InOut	MESSAGE	Instance for the internally utilized MSG command for writing a sensor parameter. This parameter must be configured as described in chapter 3.3.2.
stData	InOut	SICK_AFX60_DATA	This parameter uses the structure of the supplied SICK_AFX_DATA UDTs. The parameter values that have been read-out and are pending writing are stored here. The UDT must be instanced as Controller Tag and be transferred to the routine.
bRead	InOut	BOOL	All of the selected device parameters in data type "stData" (sub-structure GetData.Selection) are read out via a rising edge.
bWrite	InOut	BOOL	All of the selected parametric values are written to the encoder via a rising edge in data type "stData" (sub-structure SetData.Selection).
bReadDone	Output	BOOL	Indicates whether the read command has been successfully completed. TRUE: Successfully complete FALSE: Processing not complete
bWriteDone	Output	BOOL	Indicates whether the write command has been successfully completed. TRUE: Successfully complete FALSE: Processing not complete

Parameter	Declaration	Data Type	Description
bReadError	Output	BOOL	The error bit indicates whether or not a module error has occurred during processing of the read command. TRUE: no error FALSE: aborted with error
bWriteError	Output	BOOL	The error bit indicates whether or not a module error has occurred during processing of the write command. TRUE: no error FALSE: aborted with error
iReadErrorcode	Output	DINT	Read command error status (see error codes)
iWriteErrorcode	Output	DINT	Write command error status (see Error Codes)
EnableOut	Output	BOOL	Enable output (only for ladder logic)

Table 1: Parameter overview for SICK_AFX60 AOI

5 Error Codes

The output parameters iReadErrorcode and iWriteErrorcode contain the following error information in relation to the respective read or write commands:

- Module error codes
- Message error codes
- Extended message error code

Error code (32Bit)	
Extended message error codes (16Bit)	Message error codes (16Bit) Module error codes (16Bit)

Error code	Brief description	Description
16#**0000_0000**	No error	No error
16#**0000_F001**	Timeout error	The selected module command could not be executed within the timeout period.
16#**0000_F002**	Internal error	Internal module error
16#**0000_F003**	Invalid attribute read out	Device query does not correspond to the device answer. Possible cause is multiple use of the utilized MSG instance.
16#**0000_F004**	Invalid class read out	Device query does not correspond to the device answer. Possible cause is multiple use of the utilized MSG instance.
16#**0000_F005**	Invalid parameter selection	No parameters selected for read/write. Please select one or more parameters from the respective "Selection" structure.
16#**0000_F006**	Invalid input value: PositionLowLimit	Invalid input value when writing parameter "PositionLowLimit". Valid range: [0 .. 1073741823]
16#**0000_F007**	Invalid input value: PositionHighLimit	Invalid input value when writing parameter "PositionHighLimit". Valid range: [0 .. 1073741823]
16#**0000_F008**	Invalid input value: MinVelocitySetpoint	Invalid input value when writing parameter "MinVelocitySetpoint". Valid range: [0 .. 1073741823]

Error code	Brief description	Description
16#**0000_F009**	Invalid input value: MaxVelocitySetpoint	Invalid input value when writing parameter "MaxVelocitySetpoint". Valid range: [0 .. 1073741823]
16#**0000_F00A**	Invalid input value: TemperaturValueFormat	Invalid input value when writing parameter "TemperaturValueFormat". Permissible values: [4608, 4609]
16#xxxx_**0001** - 16#xxxx_**7000**	Message error	Error code generated from the utilized MSG boxes. Please refer to the RSLogix5000 help system (keyword: Error codes, message) for a complete description of these error codes.
16#**0001**_xxxx - 16#**00FF**_xxxx	Extended message error	Extended error code generated from the utilized MSG boxes. Please refer to the RSLogix5000 help system (keyword: Error codes, message) for a complete description of these error codes.

Table 2: SICK_AFX60 AOI error codes

6 Example

The following example demonstrates the procedure for handling the SICK_AFX60 AOI. Prerequisite for communication between the encoder and PLC is correct hardware configuration as well as valid parameter activation of the AOI to be called.

6.1 Program Call

When calling the program, it must be ensured that the parameters "GetMessage" and "SetMessage" are additionally configured (see chapters 3.3.1 and 3.3.2).

The timeout is given as 5000ms. This means that the routine triggers a timeout error in the event that the processing time for the module is longer than 5 seconds.

A read / write command is executed if the respective bit "bStartRead" / "bStartWrite" is triggered with a rising edge. Read and write commands are independent of each other but can also be started simultaneously.

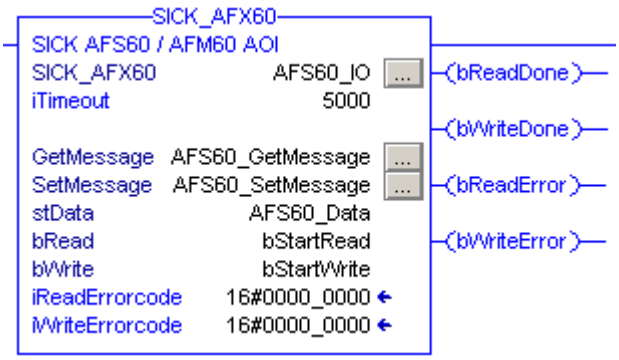

Figure 11: Program call of SICK_AFX60 routine

6.2 Reading Device Parameters

All of the preselected sensor parameters are read-out via a rising edge (bStartRead). The selection of the device parameters to be read out is undertaken in data structure "AFS60_Data.GetData.Selection" (here the selection for all supported parameters is shown).

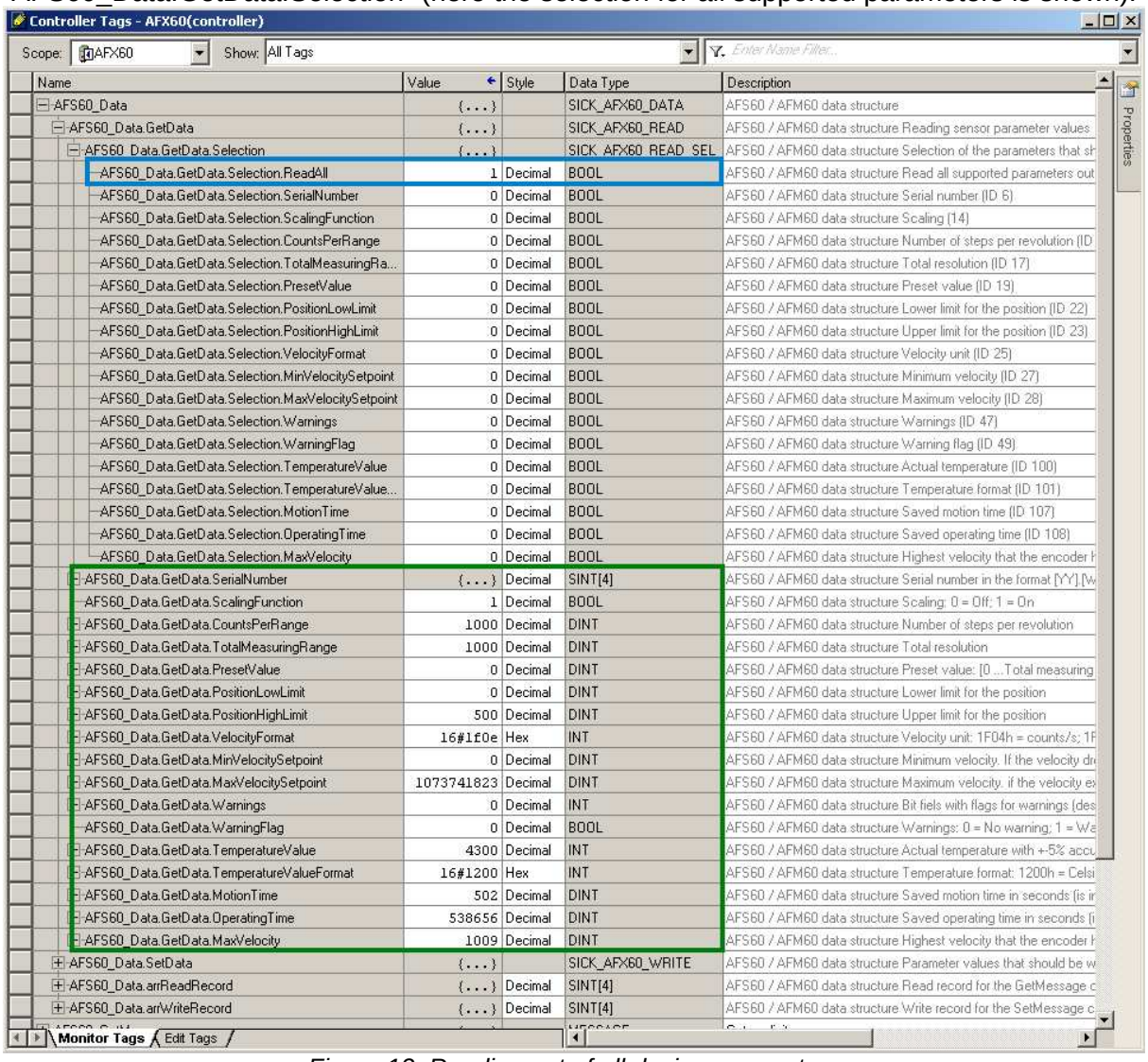

Figure 12: Reading out of all device parameters

A rising edge of output parameter "bReadDone" signalizes that the read command has been successfully completed. The corresponding values of the device parameters are stored in data structure "AFS60_Data.GetData".

6.3 Writing Device Parameters

All of the preselected parameters are written to the device via a rising edge (bStartWrite). The selection of the device parameters to be written is undertaken in data structure "AFS60_Data.SetData.Selection" (PresetValue and PositionHighLimit are shown here).

The respective value of the device parameter is defined in structure "AFS60_Data.SetData" (PresetValue:= 123 and PositionHighLimit:= 800 shown here).

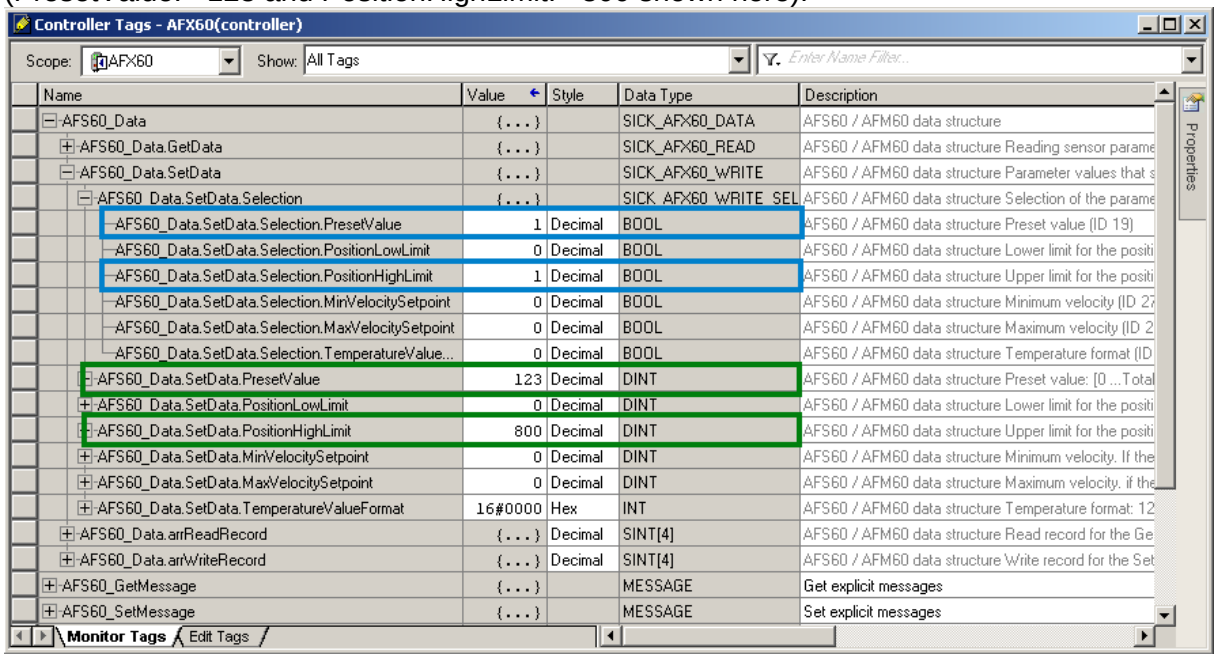

Figure 13: Selection of device parameters to be written

A rising edge of output parameter "bWriteDone" signalizes that the write command has been successfully completed.