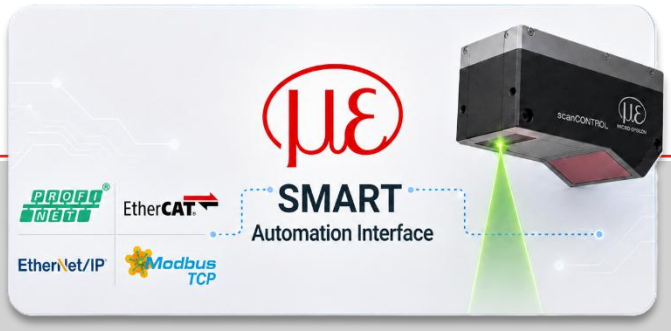
MICRO-EPSILON

Instruction Manual



Micro-Epsilon SMART Automation Interface 2.2

MICRO-EPSILON MESSTECHNIK
GmbH & Co. KG
Königbacher Straße 15

D-94496 Ortenburg / Germany

Tel. +49 (0) 8542 / 168-0
Fax +49 (0) 8542 / 168-90
e-mail: info@micro-epsilon.de
www.micro-epsilon.com

SMART Automation Interface // Documentation

Contents

Smart Automation Interface – Download Package

I. General Description

The MEAutomationCore enables automated control of MICRO-EPSILON SMART sensors via industrial communication interfaces. Measurements can be triggered via the interface, measured values can be transmitted, parameter sets can be loaded, resets can be performed, and additional control functions can be used.

The MEAutomationCore describes the control sequence on the sensor side. This includes, in particular, the Acquisition and Evaluation state machines.

The SMART Automation Interface describes the standardized interface between the MEAutomationCore and the external communication partner, for example a PLC, a PC, a Modbus client, or a fieldbus controller.

Depending on the system environment, the connection can be established via the following interfaces:

Modbus TCP, **PROFINET, EtherCAT**, **EtherNet/IP**

This documentation is intended for PLC programmers, integrators, and software developers who want to integrate a MICRO-EPSILON SMART sensor into an automation environment.

The interface is based on a defined signal sequence. The MEAutomationCore passes through various states within a state machine. These states must be passed through in a defined order. The communication partner, for example a PLC or a Modbus client, must evaluate these states and set the corresponding control signals.

This creates a robust and traceable communication sequence between the MEAutomationCore and the controller. For a stable connection, it is recommended that a dedicated state machine is also implemented on the client side.

The required settings are made via 3DInspect. In particular, this comprises preparing the parameter sets, selecting the operating mode, and setting up interface-specific settings such as endianness.

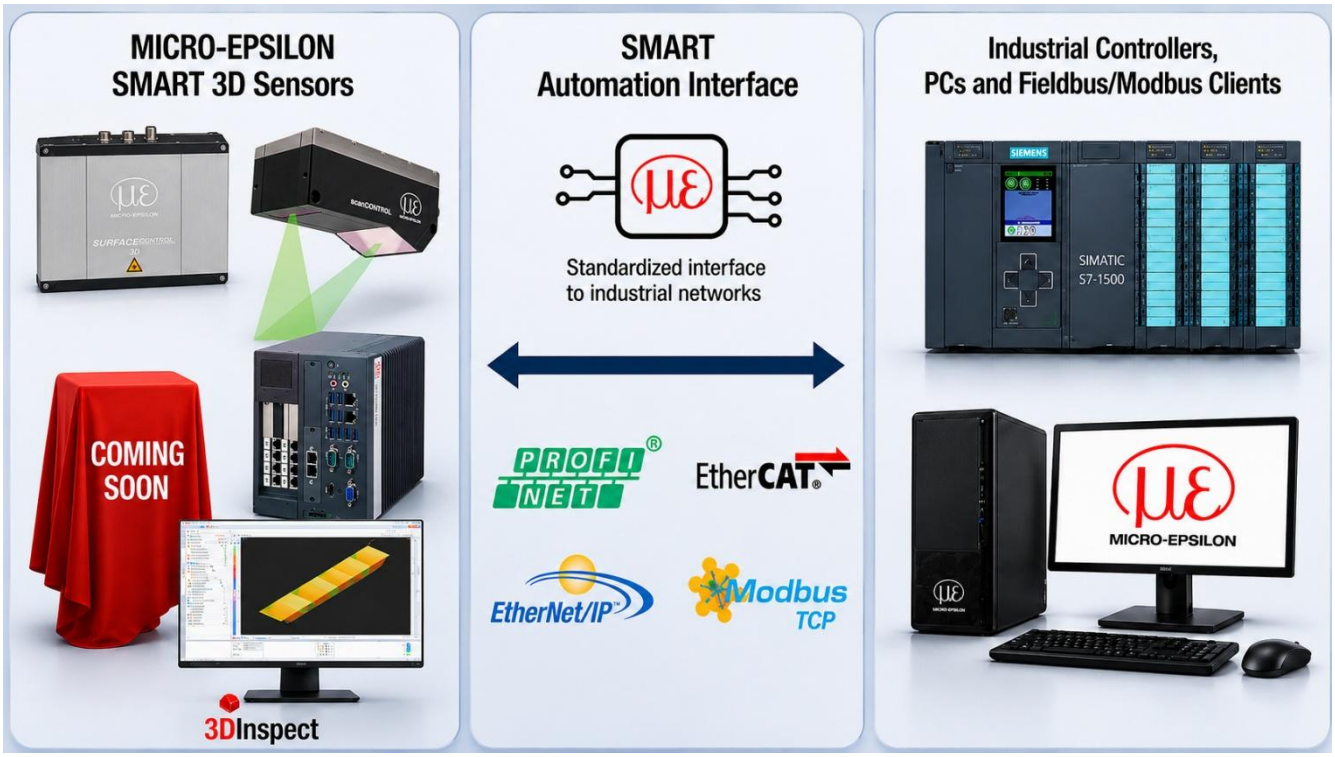


Figure 1: SMART Automation Interface

II. Setup in 3DInspect

Before using the MEAutomationCore, the sensor must be prepared in 3DInspect for automated operation. The exact configuration depends on the application, the sensor type, and the communication interface used.

1. Preparing the Measurement Task

Before starting in automated operation, the measurement task must be fully set up and tested in 3DInspect.

This includes, in particular:

- Acquisition and evaluation parameters,
- Required result values,
- Prepared parameter sets that are initially saved on the PC
- Required interface settings.

Details on parameterizing the measurement task in 3DInspect can be found in the respective 3DInspect documentation.

2. Selecting the Interface and Assigning Parameter Sets

For automated operation, the interface used and the associated settings must be selected in 3DInspect first. Second, the required parameter sets are assigned. Later, the client specifies the number of the UserSet / Recipe to be loaded via the MEAutomationCore.

The settings are made in the "Output results" view under the "Input/Output" tab.

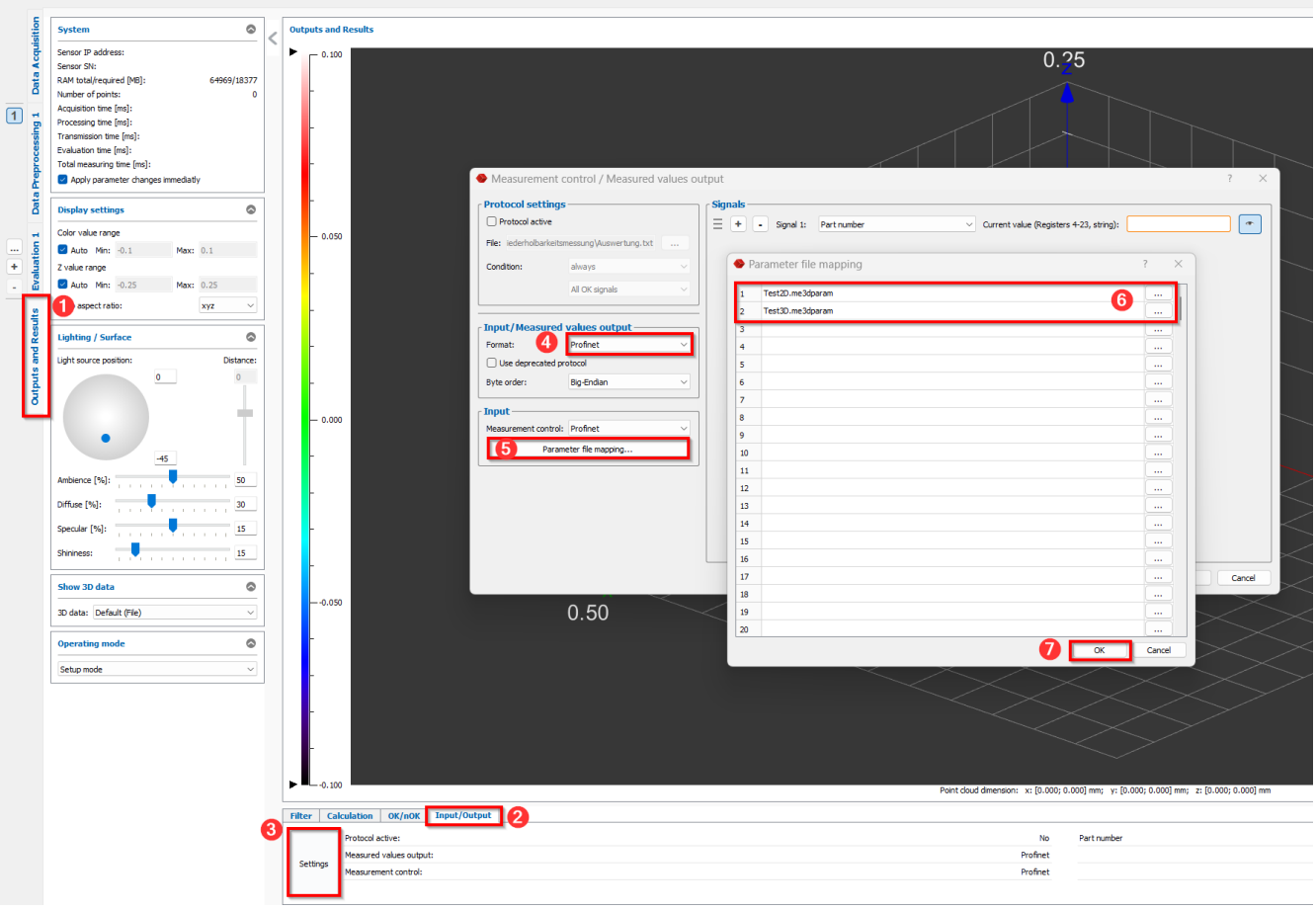


Figure 2: 3DInspect

Procedure:

- 1. Open the "Output results" view.
- 2. Select the "Input/Output" tab.
- 3. Open the settings.
- 4. In the "Input/Output measured values" area, select the desired format, for example Modbus, EtherCAT, EtherNet/IP, or PROFINET.
- 4.1 In the "Input" area under "Measurement control", select the network interface used. The "Sensor HW trigger" option is not supported for control via the MEAutomationCore.
- 4.2 For fieldbus connections, set the appropriate byte order:
    - PROFINET: Default Big-Endian
    - EtherCAT: Default Little-Endian
    - EtherNet/IP: Default Little-Endian
- 4.3 For Modbus, check the Modbus port. Port 502 is used by default.
- 5. Select the "Assign parameter files..." button.
- 6. Assign the desired parameter files to the corresponding numbers.
- 7. Confirm the assignment with "OK".

The number of the respective row corresponds to the number that is later specified by the client. This allows the MEAutomationCore to load the corresponding UserSet/Recipe.

The byte order must match the setting on the client side. If it is set incorrectly, status values, error codes, or measurement results will be interpreted incorrectly.

3. Setup Mode - Measurement Mode

For sensor operation, a distinction is made between setup mode and measurement mode.

In setup mode, the sensor is configured and tested in 3DInspect. In this state, parameters can be adjusted, measurement tasks can be checked, and parameter sets can be prepared.

For automated operation, 3DInspect must then be switched to measurement mode. Only in measurement mode can the sensor be controlled via the MEAutomationCore by the connected client.

Now the client can request the automatic mode, load parameter sets, start measurements, read results, or acknowledge errors.

The control via the MEAutomationCore is only possible if the sensor has been set up in 3DInspect, if no active error is present, and if the fieldbus or Modbus communication to the sensor is established.

III. Implemented Functions and Signals

The following functions and signals are provided via the SMART Automation Interface for controlling the MEAutomationCore. Depending on the communication type, these signals can be mapped via Modbus TCP, PROFINET, EtherCAT, or EtherNet/IP.

The term client describes the communication partner of the MEAutomationCore, for example a PLC, PC, Modbus client, or fieldbus controller.

1. Control and Setpoint Signals

Monitoring (bMonitoring):

Activates the monitoring function. This enables the SMART sensor to send measurement data and measured values to 3DInspect for visualization.

Select parameter set (nUserSet):

Specifies the number of the UserSet / Recipe to be loaded. The number must correspond to the parameter file assigned in 3DInspect. Up to 255 UserSets / Recipes can be loaded.

Job Sequence Number (sJSN):

Transfers a Job Sequence Number with up to 40 characters from the client to the MEAutomationCore. This enables measurement data and measurement results to be assigned to production data.

Emitter off (EmitterOff):

Switches off the sensor's light source.

Acknowledge error (ResetError):

Acknowledges a pending error via a positive edge. Error handling and error storage must be implemented by the user.

Reset (bReset):

Starts the reset routine. During this process, the interface signals are reset and the state machine is moved to a defined initial state.

Request automatic mode (bAutomaticMode):

Requests automatic mode. This is possible if 3DInspect is not in setup mode.

Automatic registration (bAutomaticRegistration):

Starts automatic registration for sensors that support a registration routine.

Start measurement (bInPosition_Start):

Triggers a measurement in discrete operation. The signal should be set when the target is in the measurement position. In discrete operation, the start signal is automatically reset after it has been accepted.

In continuous operation, the sensor starts the measurement cycles automatically as long as the start signal is present.

Flush (bFlush):

For line sensors, the flush signal terminates an ongoing 3D measurement, even if the profiles for a 3D point cloud have not yet been fully acquired.

2. Feedback Signals

Live signal (Live):

The live signal is toggled at a frequency of 2 Hz and is used to monitor communication between the MEAutomationCore and the client.

Unload part (bUnloadPart):

Is set to TRUE as the soon as exposure or the data acquisition for the measurement is complete. The target can then be transported onward while evaluation of the acquired measurement data continues.

Error code (ErrorCode):

Outputs the currently pending error code.

Acquisition status (InStateAcq):

Displays the current state of the Acquisition state machine.

Evaluation status (InStateEval):

Displays the current state of the Evaluation state machine.

Active UserSet / Recipe (UserSet):

Outputs the UserSet / Recipe currently loaded on the sensor.

Validated results (Results):

Provides the validated measurement results for further processing.

3. Data Areas

Basic input data (INBasis):

Contains basic feedback from the MEAutomationCore, for example status information, live signal, and error states.

Smart input data (INSmart):

Contains extended feedback from the MEAutomationCore, for example states of the state machines.

Unvalidated results (INResults):

Contains the result data provided by the MEAutomationCore.

Basic output data (OUTBasis):

Contains basic control commands that are transmitted from the client to the MEAutomationCore.

Smart output data (OUTSmart):

Contains extended control commands, for example automatic mode, start signal, flush, and result acknowledgment.

Job Sequence Number (OUTJSN):

Transmits the Job Sequence Number for assigning the measured values to the production data.

IV. Integration via Modbus TCP

To use the SMART Automation Interface via Modbus TCP, a TCP connection to port 502 of the sensor used must first be established.

Basic Modbus Commands

Two Modbus commands are used for the communication (index base 1):

- **03 - Read Input Registers**
  Start register: 0
  Number of registers: 128
- **16 - Write Multiple Holding Registers**
  Start register: 0
  Number of registers: 24

Cyclic Data Acquisition

During operation, it is recommended that the Read Input Registers command be executed cyclically in order to continuously detect changes in the sensor's Automation Core.

The polling frequency depends on the application:

- 3D measurement: typically 20-50 Hz
- 2D profile measurement: significantly higher measurement frequencies; here, faster polling or alternatively another transmission method for the measured values may be required.

Write Accesses

The Write Multiple Holding Registers command is executed only when required, for example for parameterization or control of the sensor.

Example Implementation

An example implementation for Modbus/TCP communication in Python is available on the Micro-Epsilon website in the form of the "Micro-Epsilon Modbus Tool". Through its clear visualization of the state machine and the transmitted parameters, the tool provides a quick and intuitive introduction to understanding Modbus communication.

After starting the tool, a ZIP-compressed folder containing the Python source code can be downloaded in the "Terminal example" tab via the "Save source code" button. This source code represents the basic structure of communication with the MEAutomationCore via Modbus. The package also contains example implementations showing, for example, how the MEAutomationCore can be controlled via the command line or how an initial measurement can be performed.

This also makes the Modbus tool a compact introductory tutorial for Python programming in the context of Modbus communication.

Schematic Sequence

It is recommended that the following sequence be followed in the client program to enable smooth communication with the MEAutomationCore:

1. Read the Modbus response to the previously sent command.
2. Run through the Acquisition state machine on the client side and set the required interface signals.

   → If write access to holding registers is required, note this requirement.
3. Run through the Evaluation state machine on the client side and set the required interface signals.

   → If write access to holding registers is required, also note this requirement.
4. Check the noted requirement:

   → If write access is required, perform Write Holding Registers.

→ Otherwise, perform Read Input Registers again.

Register Assignments/Interface Signals

The following table shows the signals of the SMART Automation Interface that are required for controlling the MEAutomationCore. From the perspective of the communication partner/client, the holding registers are the output data that is sent to the sensor. The input registers represent the input signals that are received.

		Holding-Register	Bit	Signal	Meaning	Input-Register	Bit	Signal	Meaning
Basis		1	0	Reserved		1	0	I_Live	Live bit
		1	1	Q_ResultsDisable	Gate for measurement output	1	1	Reserved	
		1	2	Q_EmitterOff	Switches off e.g. light source	1	2	I_EmitterOff	State of e.g. Light Source
		1	3	Q_Reset	Start measurement; part in position	1	3	Reserved	
		1	4	Q_ExecuteCalibration	Master, registration, referencing	1	4	I_CalibrationSuccess	Indicates successful calibration
		1	5	Q_AcquisitionDisable	Gate for raw data acquisition	1	5	Reserved	
		1	6	Q_InitCounters	Initialize counters, e.g. Encoder	1	6	Reserved	
		1	7	Q_CalibrationDisable	Deactivates the calibration/registration	1	7	I_CalibrationOn	Indicates calibration active/inactive
		1	8	Q_ResetError	Resetting the error word	1	8	Reserved	
		1	9	Reserved		1	9	Reserved	
		1	10	Reserved		1	10	Reserved	
		1	11	Reserved		1	11	I_IO1	State of hardware input 1
		1	12	Reserved		1	12	I_IO2	State of hardware input 2
		1	13	Reserved		1	13	I_IO3	State of hardware input 3
		1	14	Reserved		1	14	I_IO4	State of hardware input 4
		1	15	Reserved		1	15	Reserved	
		2	0-7	Reserved		2	0-7	IB_StateAcquisition	Current status of the statemachine acquisition
		2	8-15	QB_UserSet	UserSet to be loaded	2	8-15	IB_UserSet	Currently loaded UserSet
		3	0-15	Reserved		3	0-15	IW_ErrorCode	
Smart		4	0	Q_AutomaticMode	0: Manual; 1: Automatic	4	0	I_StateMachineMode	0: Discrete; 1: Cont
		4	1	Q_Start	Start 3D measurement; part in position	4	1	Reserved	
		4	2	Q_Flush	Flush/abort measurement (line sensors)	4	2	Reserved	
		4	3	Q_ResultsAck	Acknowledge measurement results	4	3	Reserved	
		4	4	Q_MonitoringOn	Activate streaming on GenICam channel	4	4	Reserved	
		4	5	Reserved		4	5	Reserved	
		4	6	Reserved		4	6	Reserved	
		4	7	Q_SingleTrigger	Start single measurement (e.g. profile trigger)	4	7	Reserved	
		4	8-15	Reserved		4	8-15	IB_StateEvaluation	Current status of the statemachine evaluation
JSN	Results	5-24		Q_JSN	Job sequence number String to identify the measurement	5-124		I_Results	

Figure 3: MEAutomationInterface Signals

V. Integration via Fieldbuses

The SMART Automation Interface can be integrated into a fieldbus environment via PROFINET, EtherCAT, or EtherNet/IP. Data exchange takes place between the client and the MEAutomationCore via the respective fieldbus.

The necessary files for integration are available for each fieldbus. These include the respective device description file, an example program with the SMART Automation Interface, and the associated SMART Automation Interface Library.

For integration, the appropriate device description file is required in the respective engineering tool:

- PROFINET: GSDML file, e.g. for Siemens TIA Portal
- EtherCAT: ESI file, e.g. for TwinCAT
- EtherNet/IP: EDS file, e.g. for Allen-Bradley

The interface used and the appropriate byte order (endianness) must first be set in 3DInspect. The procedure is described in Chapter II, Section 2, "Selecting the interface and assigning parameter sets".

1. Interface Signals

The interface signals of the SMART Automation Interface have the same structure for all fieldbuses. The direction of the data is described from the perspective of the client or controller.

The output data is transmitted from the client to the MEAutomationCore. The input data is transmitted from the MEAutomationCore to the client.

The exact division of the data areas and the assignment of the individual signals are shown in the following figure.

	Output Adress		Signal	Meaning	Input Adress	Bit	Signal	Meaning
Basis	0.0	Bit	Reserved		0.0	Bit	I_Live	Live bit
	0.1	Bit	Q_ResultsDisable	Gate for measurement output	0.1	Bit	Reserved	
	0.2	Bit	Q_EmitterOff	Switches off e.g. light source	0.2	Bit	I_EmitterOff	State of e.g. Light Source
	0.3	Bit	Q_Reset		0.3	Bit	Reserved	
	0.4	Bit	Q_ExecuteCalibration	Master, registration, referencing	0.4	Bit	I_CalibrationSuccess	Indicates successful calibration
	0.5	Bit	Q_AcquisitionDisable	Gate for raw data acquisition	0.5	Bit	Reserved	
	0.6	Bit	Q_InitCounters	Initialize counters, e.g. Encoder	0.6	Bit	Reserved	
	0.7	Bit	Q_CalibrationDisable	Deactivates the calibration/registration	0.7	Bit	I_CalibrationOn	Indicates calibration active/inactive
	1.0	Bit	Q_ResetError	Resetting the error word	1.0	Bit	Reserved	
	1.1	Bit	Reserved		1.1	Bit	Reserved	
	1.2	Bit	Reserved		1.2	Bit	Reserved	
	1.3	Bit	Reserved		1.3	Bit	I_IO1	State of hardware input 1
	1.4	Bit	Reserved		1.4	Bit	I_IO2	State of hardware input 2
	1.5	Bit	Reserved		1.5	Bit	I_IO3	State of hardware input 3
	1.6	Bit	Reserved		1.6	Bit	I_IO4	State of hardware input 4
	1.7	Bit	Reserved		1.7	Bit	Reserved	
	2	Byte	Reserved		2	Byte	IB_StateAcquisition	Current status of the statemachine acquisition
	3	Byte	QB_UserSet	UserSet to be loaded	3	Byte	IB_UserSet	Currently loaded UserSet
	4	Word	Reserved		4	Word	IW_ErrorCode	
Smart	6.0	Bit	Q_AutomaticMode	0: Manual; 1: Automatic	6.0	Bit	I_StateMachineMode	0: Discrete; 1: Cont
	6.1	Bit	Q_Start	Start measurement; part in position	6.1	Bit	Reserved	
	6.2	Bit	Q_Flush	Flush/abort measurement (line sensors)	6.2	Bit	Reserved	
	6.3	Bit	Q_ResultsAck	Acknowledge measurement results	6.3	Bit	Reserved	
	6.4	Bit	Q_MonitoringOn	Activate streaming on GenICam channel	6.4	Bit	Reserved	
	6.5	Bit	Reserved		6.5	Bit	Reserved	
	6.6	Bit	Reserved		6.6	Bit	Reserved	
	6.7	Bit	Q_SingleTrigger	Start single measurement (e.g. profile trigger)	6.7	Bit	Reserved	
	7	Byte	Reserved		7	Byte	IB_StateEvaluation	Current status of the statemachine evaluation
JSN Results	8-47	40 Bytes	Q_JSN	Job sequence number String to identify the measurement	8 - 127	120 Bytes	I_Results	

Figure 4: MEAutomationInterface Signals

2. PROFINET

Provide and Import the GSDML File

In TIA Portal, open the dialog Options -> Manage device description files (GSD). Select the folder in which the GSDML file is stored, select the file, and click "Install".

Note: A detailed description of how to install GSD files is provided in the Siemens documentation:

https://support.industry.siemens.com/cs/document/109738401/how-do-you-install-a-gsd-file-and-which-gsd-file-version-is-released-for-which-version-of-tia-portal-?dti=0&dl=en&lc=de-DE

Add Hardware from the Catalog

In the project under Devices & networks, open the hardware catalog. Path:

Other field devices -> PROFINET IO -> Sensors -> MICRO-EPSILON MESSTECHNIK -> [Product family] -> [Sensor]

Add the device to the network view by drag and drop and configure it.

Set Parameters and Check the Connection

The IP address of the sensor must be in the same subnet as the controller. In addition, a unique device name must be assigned. Special characters and spaces should be avoided.

The project is then compiled and loaded into the controller. After establishing the online connection, the device must be displayed without errors. Cyclic communication can be checked using the toggling live bit.

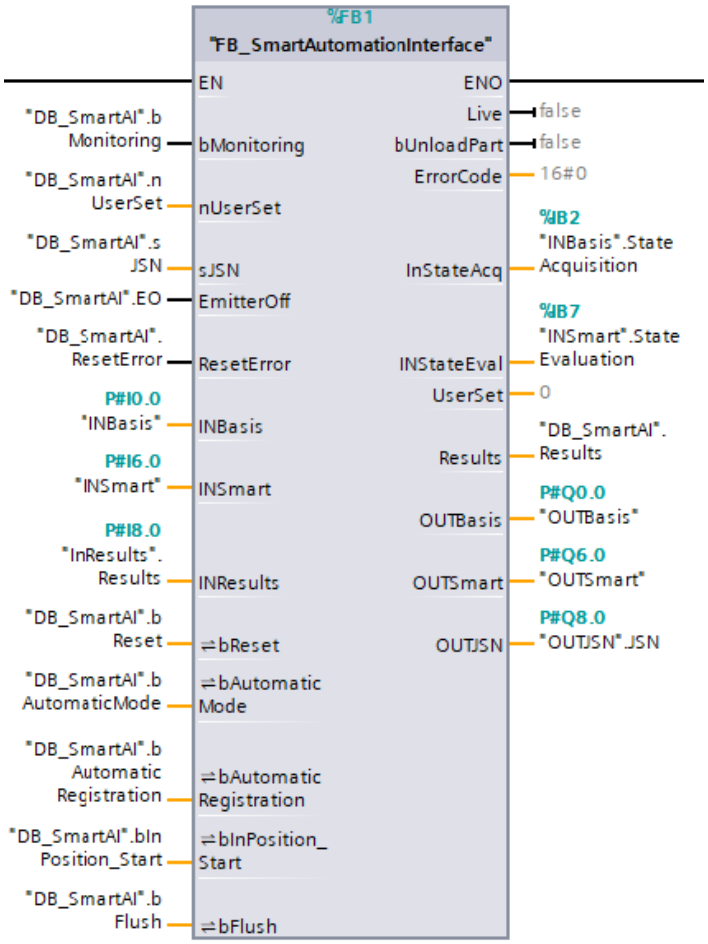


Figure 5: PROFINET Integration

3. EtherCAT

Provide and Import the ESI File

Copy the ESI file to the EtherCAT directory of TwinCAT. Default path:

C:\TwinCAT\3.1\Config\Io\EtherCAT

Then start TwinCAT 3 so that the device description is read.

Add Hardware from the Catalog

In the project tree under I/O -> Devices, execute the Scan command.

Accept the EtherCAT device found and confirm the subsequent search for boxes with "Yes". The device is then automatically added to the EtherCAT topology and can be configured.

Assign Variables

The input and output variables of the PLC project must be assigned to the interface signals of the SMART Automation Interface.

To do this, select the relevant variables and link them to the corresponding hardware channels using multiple linking.

Ensure the correct assignment and order of the input and output data.

Check the Connection

Activate the configuration and establish the online connection to the EtherCAT system.

The live bit can be used to check communication. When the live bit toggles, data is transferred cyclically between the sensor and the client.

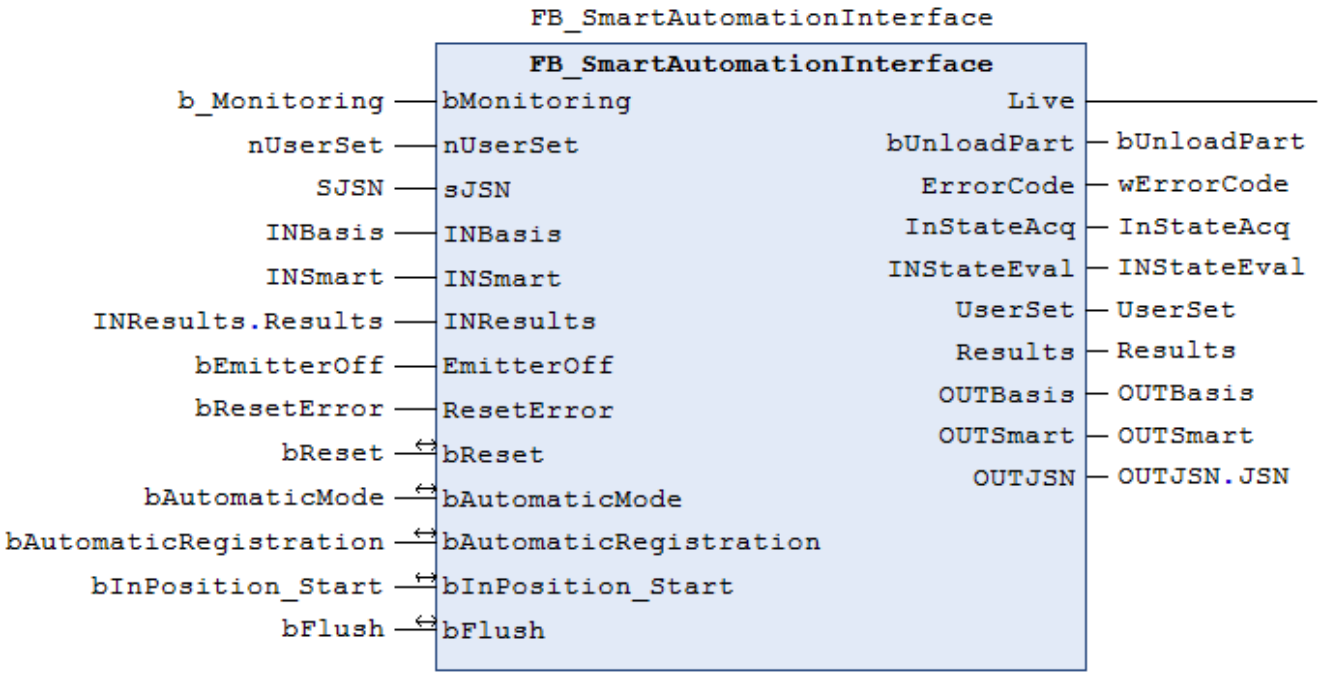


Figure 6: EtherCAT Integration

4. EtherNet/IP

Provide and Import the EDS File

Install the EDS file belonging to the device used in the Rockwell engineering environment, for example via the EDS Hardware Installation Tool. After successful installation, the device is available in the hardware catalog.

Assign IP Address

The IP address of the EtherNet/IP participant must be in the same subnet as the controller. During initial commissioning, the address can be assigned, for example, with the Rockwell BOOTP/DHCP tool via the MAC address of the device.

After assigning the address, BOOTP/DHCP must be disabled or the address mode must be changed to a static IP address. The device should then be restarted and checked to ensure that the IP address is retained.

Add Hardware from the Catalog

In the Studio 5000 project, open the I/O configuration of the Ethernet scanner or Ethernet module used. Add the device from the hardware catalog and enter the previously assigned IP address.

The IP address entered in the project must match the IP address set on the EtherNet/IP participant.

Check the Connection

Load the project into the controller and establish the online connection. The device must be displayed without errors. Cyclic communication with the MEAutomationCore can be checked using the live bit. When the live bit toggles, data is transferred cyclically between the MEAutomationCore and the controller.

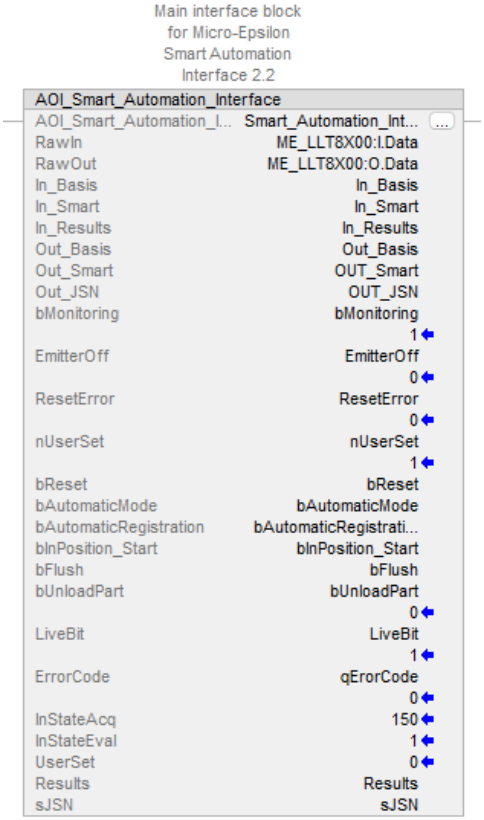


Figure 7: EtherNet/IP Integration

VI. Basic Sequence up to the First Measurement

Prerequisite:

Communication with the MEAutomationCore has been established, no active error is present, and the live bit is toggling.

1. Check the basic state

Wait until the MEAutomationCore has reached Acquisition State 150. In this state, the system is in Manual Mode.

2. Specify UserSet / Recipe

Specify the desired UserSet / Recipe. The UserSet / Recipe is loaded by the MEAutomationCore after AutomaticMode has been activated.

3. Activate automatic mode

Activate AutomaticMode.

4. Wait for the loading process

During loading, the MEAutomationCore changes to Acquisition State 200 to load the UserSet / Recipe.

5. Check measurement readiness

After successful loading, the MEAutomationCore changes to Acquisition State 1 (Ready). In this state, the system is ready for measurement.

6. Start measurement

As soon as I_StateAcq = 1 is present, the measurement can be triggered via the start signal bInPosition_Start.

VII. Appendix

1. Appendix A: Error Coding ErrorCode Field:

Code	Meaning	Description
0	OK	No error
1	HARDWARE_ERROR	Connection to sensor not possible. Connection to sensor lost.
2	DESCRIPTION_FILE_ERROR	Description file loade error, e.g. ProfileUnit configuration error.
3	PARSING_ERROR	Error while parsing a file.
4	HARDWARE_INIT_ERROR	Error at initializing hardware.
5	INIT_MEASUREMENT_ERROR	Error while initializing the measurement, e.g. while homing.
6	MEGS_ERROR	Error start/init MEGS.
7	INIT_STREAMING_ERROR	Error while init streaming channel to GenICam Client
8	INVALID_CONFIGURATION	Invalid configuration of setting on the Sensor
100	ARM_ACQUISITION_ERROR	Error while arming the acquisition.
101	START_ACQUISITION_ERROR	Error while starting the acquisition.
102	TRIGGER_SW_ERROR	Error while triggering via software.
103	STOP_ACQUISITION_ERROR	Error while stopping the acquisition.
200	EVALUATION_ERROR	Error while evaluating the point cloud.
201	CALIBRATION_FILE_ERROR	Error during registration/calibration.
202	USER_SET_NUMBER_NOT_REFERENCED	Error while loading the user set. The user set is not defined on the sensor.
203	TIMEOUT_LOADING_USERSET	Loading the user set took to long.
204	LOADING_RECIPE_ERROR	Error while loading the recipe file.
900	SENSOR_LOST	Connection to the Sensor lost.
901	HARDWARE_OVERHEATING	Hardware overheat error.
902	PLC_DISCONNECTED	Lost connection to the PLC.
903	SETUP_MODE_ACTIVE	Sensor in setup mode, no control over automation interface possible.
32767	GENERAL_ERROR	Error not defined.
65520 - 65535	Application specific error	Can be defined by the sensor application.

Figure 8: Error Coding

2. Appendix B: Example State Machine of the Communication Partner

It is recommended that the sequence control on the side of the communication partner also be implemented as a state machine. The communication partner may be, for example, a PLC, a Modbus master, a PC program, or another client.

The following diagrams show an example implementation from the perspective of the communication partner. They serve as a basis for custom implementations and must be adapted depending on the application.

Representation of the Signal Sequences

The diagrams show which states are passed through and which signals are set, reset, or evaluated in the process.

The signals shown in black are exchanged directly between the communication partner and the MEAutomationCore.

The following applies:

- I_: Input signal of the communication partner, comes from the MEAutomationCore

  Example: I_StateAcq, I_StateEval, I_Results

- Q_: Output signal of the communication partner, goes to the MEAutomationCore

  Example: Q_Reset, Q_AutomaticMode, Q_UserSet, Q_Start

The signals shown in orange belong to the higher-level automation environment, for example to inputs and outputs of a PLC block. Examples are Reset, AutomaticMode, UserSet, PartInPosition, or UnloadPart.

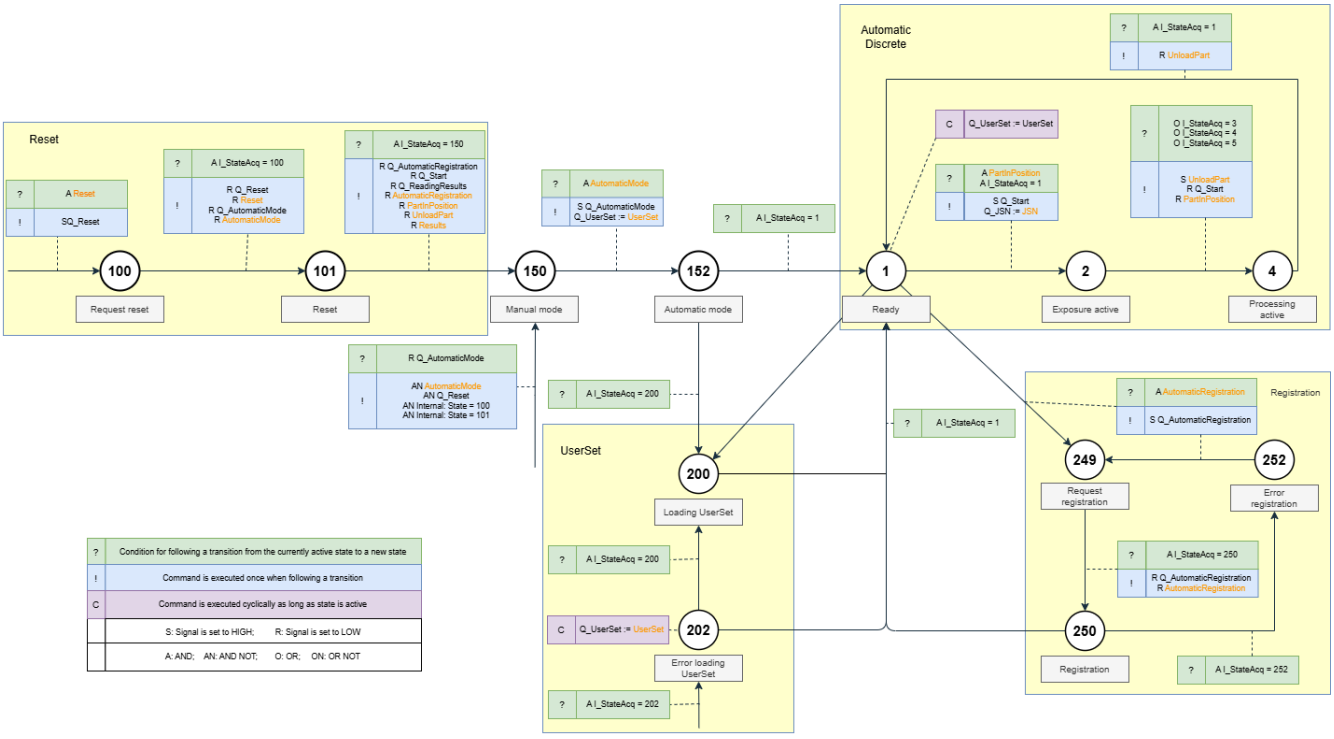


Figure 9: Example State Machine

The live bit is toggled at a frequency of 2 Hz and is used to monitor communication. If the live bit stops toggling, a communication fault must be assumed.

Use Cases Described Below

The following sections describe typical signal sequences:

- Reset

- Switching between Manual Mode and Automatic Mode
- Loading a UserSet / Recipe
- Triggering a registration
- Measurement in discrete operation
- Evaluation in discrete operation
- Measurement in continuous operation
- Sensor-side State Machines of the MEAutomationCore

Reset Sequence from the Perspective of the Communication Partner

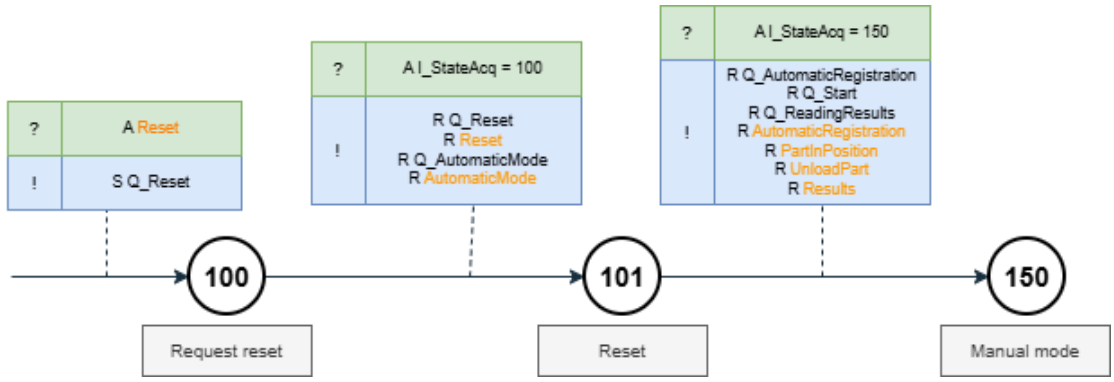


Figure 10: Reset Sequence from the Perspective of the Communication Partner

Communication partner

When the reset request is triggered by the automation environment, the output Q_Reset is set and the state changes to state 100.

The communication partner reads this status, thereby confirming that the reset command has been received. It can then reset its Q_Reset and Reset signals and changes to state 101. In addition, Q_AutomaticMode is reset to ensure that the MEAutomationCore remains in ManualMode and does not simply pass through the state immediately.

When the communication partner reads the status I_StateAcq = 150, it knows that the MEAutomationCore has completed the reset sequence and also resets its relevant signals in order to switch to Manual Mode.

MEAutomationCore

This signal is read by the MEAutomationCore and, if it is not already in state 100, it changes to this state.

The MEAutomationCore, detects that the reset signal has been reset and then changes to state 101. It also resets its relevant interface signals. Once the internal initialization processes have been completed, the MEAutomationCore changes to state 150.

This behavior applies to both continuous and discrete operation. In the discrete case, the Evaluation state machine must also be considered.

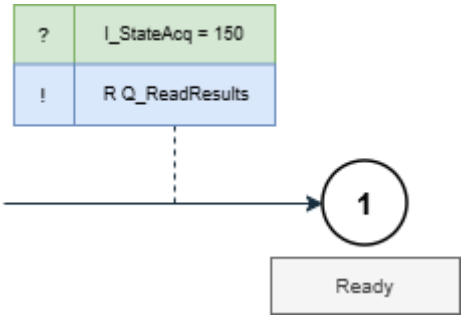


Figure 11: Reset Signal Sequence of the Communication Partner's Evaluation State Machine

Because Manual Mode is always passed through during a reset, the communication partner checks for I_StateAcq = 150. If this is present, the Q_ReadResults signal is reset and the sequence jumps back to Evaluation State 1.

Changing Operating Mode: Manual Mode / Automatic Mode

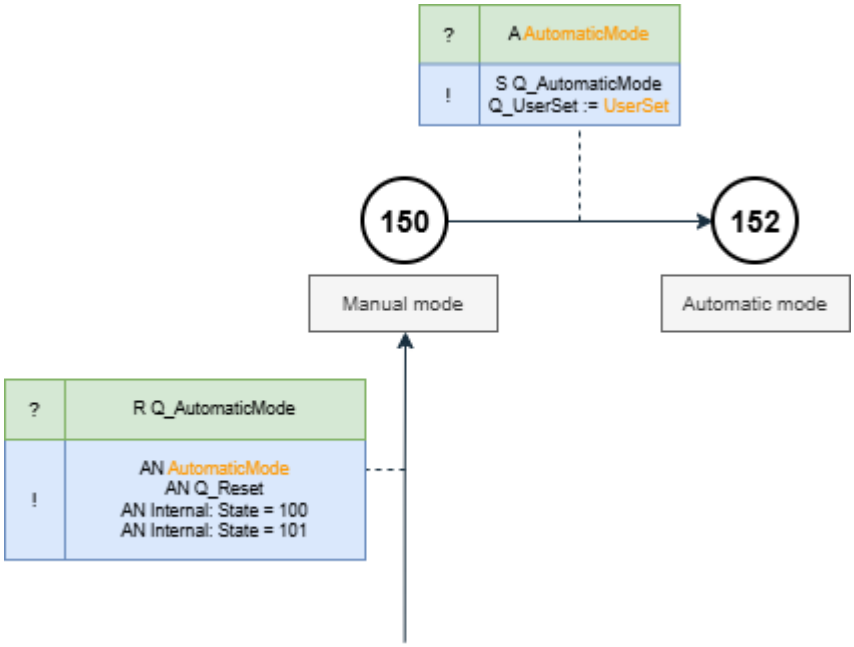


Figure 12: Switching the Operating Mode in the Acquisition State Machine from the Perspective of the Communication Partner

Communication partner

If AutomaticMode is to be reset and the state machine is not currently in a reset routine or a reset has not been requested, the output Q_AutomaticMode is reset and the state changes to state 150 "Manual Mode".

The state machine remains in Manual Mode until a request for AutomaticMode is triggered again by the automation environment.

MEAutomationCore

If the control bit for AutomaticMode has been reset and the state machine is not currently in a reset routine or a reset has not been requested, the state changes to state 150 "Manual Mode". Q_ActualUserSet := 0 is also set. This ensures that after switching to AutomaticMode, the UserSet is loaded again, even if it may already have been loaded previously.

The output Q_AutomaticMode is then set and the requested UserSet is transferred to the output Q_UserSet.

Via the input I_AutomaticMode, the state machine changes to state 151 and then, as soon as the internal process for switching the UserSet is complete, to state 152 "Automatic Mode".

This behavior applies to both continuous and discrete operation. In the discrete case, the Evaluation state machine must also be considered (see Figure 10).

Load UserSet / Recipe

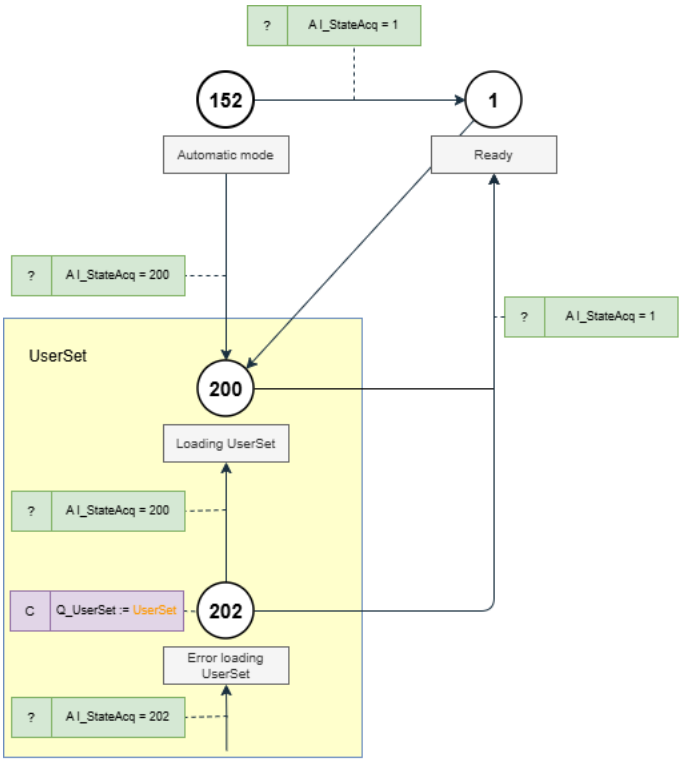


Figure 13: Loading a UserSet from the Perspective of the Communication Partner

Communication partner

When the MEAutomationCore has changed to either state 1 or state 200 after AutomaticMode, the communication partner reads back this I_StateAcq and also changes to the corresponding state.

In state 200, the sequence waits until completion of loading is detected via I_StateAcq, and then changes to the corresponding state.

MEAutomationCore

Starting from Automatic Mode, the MEAutomationCore immediately changes to "Ready" State 1 if no UserSet was transferred, or to state 200 "Load UserSet" if a UserSet was transferred when switching to Automatic Mode.

At the beginning of state 200, I_UserSet is written to Q_ActualUserSet and the loading process is performed. If it completes successfully, the state changes to "Ready" State 1. In case of an error, the state changes to state 202.

In error state 202, the sequence waits until a UserSet is transferred that is not equal to the previous UserSet and is not equal to 0. Than it changes to state 200 and attempts loading again.

In state 202, the UserSet is cyclically transferred to Q_UserSet. This causes loading to be triggered again in the MEAutomationCore when a new UserSet is transferred.
Since the transition between states 200 and 202 in the MEAutomationCore can theoretically occur so quickly that the communication partner does not detect it, the transition to state 202 in the communication partner is independent of the previous state.

When operational readiness is present in Ready "State" 1, the UserSet is cyclically transferred to Q_UserSet, enabling a new UserSet to be loaded.

From "Ready" State 1, the state can also change to state 200 if a new UserSet not equal to 0 is transferred.

This behavior applies to both continuous and discrete operation.

Trigger Registration

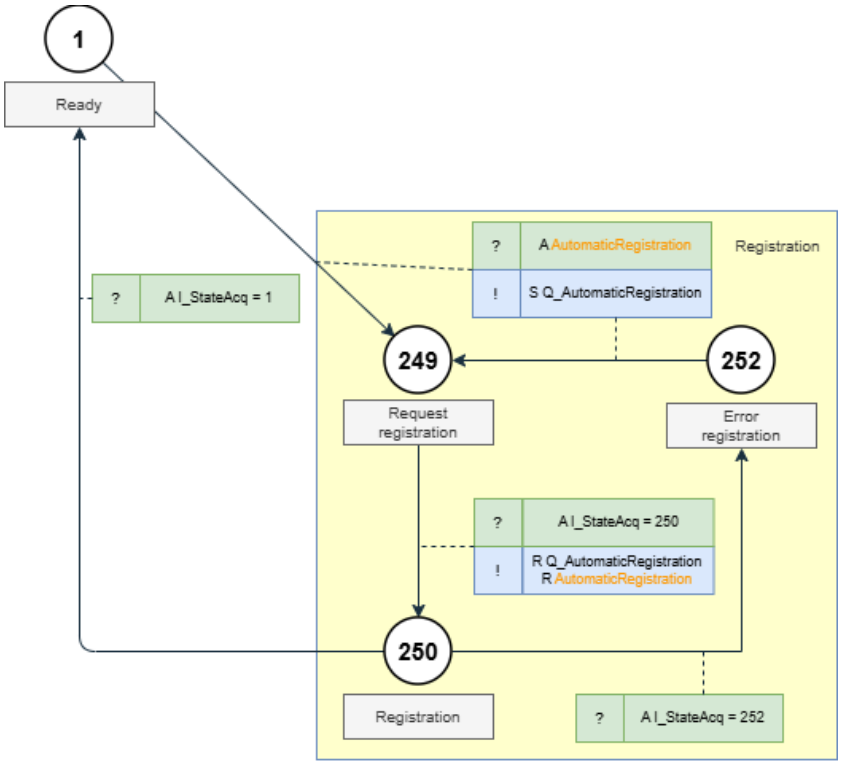


Figure 14: Triggering a Registration from the Perspective of the Communication Partner

Communication partner

When the AutomaticRegistration request is triggered by the automation environment, the output Q_AutomaticRegistration is set and the state changes to state 249.

The communication partner reads this status back, thereby confirming that the registration command has been received. Than it can reset its Q_AutomaticRegistration and AutomaticRegistration signals and changes to state 250.

MEAutomationCore

This signal is read by the MEAutomationCore, thus the MEAutomationCore changes to state 250 and starts the registration.

After completion of the registration routine, the MEAutomationCore changes back to state 1 if the routine was successful. If an error

Depending on the status of the MEAutomationCore (I_StateAcq), the communication partner also changes either to state 1 or to state 252. In the event of an error, the registration must be triggered again by the automation environment (AutomaticRegistration) and by setting the output Q_AutomaticRegistration.

This behavior applies to both continuous and discrete operation.

Measurement Sequence in Discrete Operation

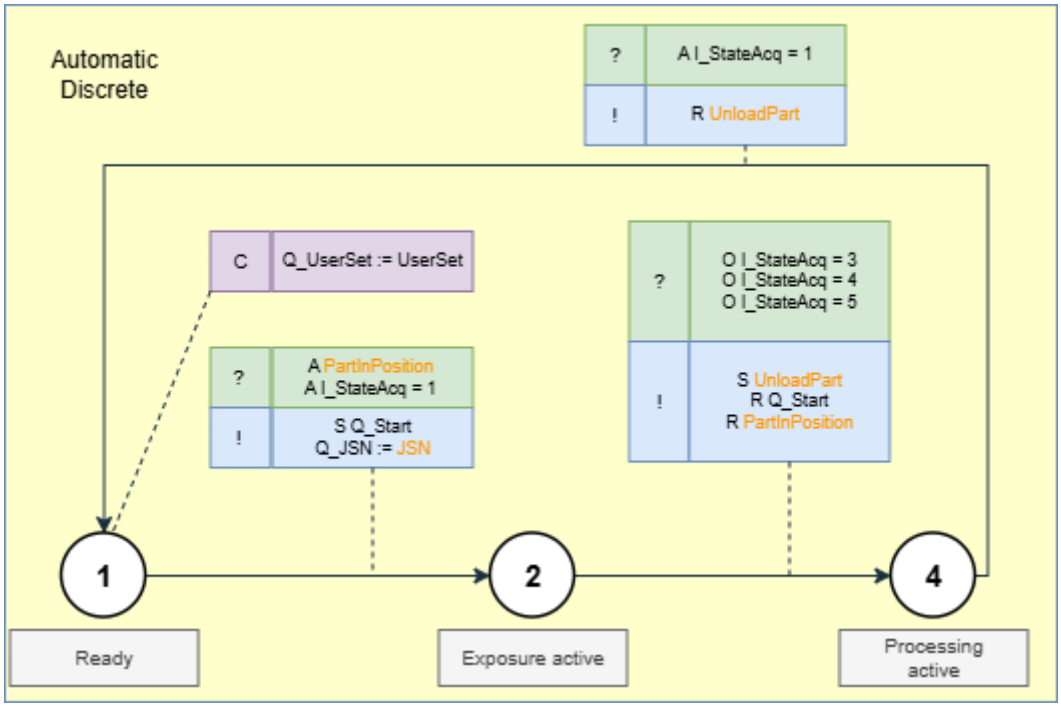


Figure 15: Measurement in the Discrete Case from the Perspective of the Communication Partner

Communication partner

In Ready State 1, if request for a measurement from the automation environment is set by PartInPosition and the MEAutomationCore is also in Ready State (I_StateAcq = 1), the communication partner sets Q_Start, writes the JSN to the respective output Q_JSN, and changes to state 2.

As soon as the communication partner reads back I_StateAcq = 3, it knows that data acquisition is complete. It can therefore set the motion release UnloadPart and reset PartInPosition and Q_Start.

MEAutomationCore

When the start signal of the communication partner is read and the currently output UserSet corresponds to the requested UserSet, the measurement is started and the state changes to state 2.

State 2 is active until data acquisition has been completed. The data is processed into a 3D point cloud in state 3 and then transferred to the Evaluation state machine (state 4). In state 5, the MEAutomationCore waits again for feedback from the communication partner.

occurs during the routine, the state changes to state 252. A prerequisite for both transitions is that the input I_PerformRegistration has already been reset.

Since the state transition from state 3 to state 5 on the MEAutomationCore side occurs without interaction from the communication partner and the states could theoretically be missed, I_StateAcq = 4 or I_StateAcq = 5 is also checked.

I_StateAcq = 1 is the transition condition for the communication partner to change to state 1. In the process, the signal to the automation environment UnloadPart is reset again.

With the start signal reset, the MEAutomationCore can return to "Ready" State 1.

In the discrete case, the Evaluation state machine must also be considered.

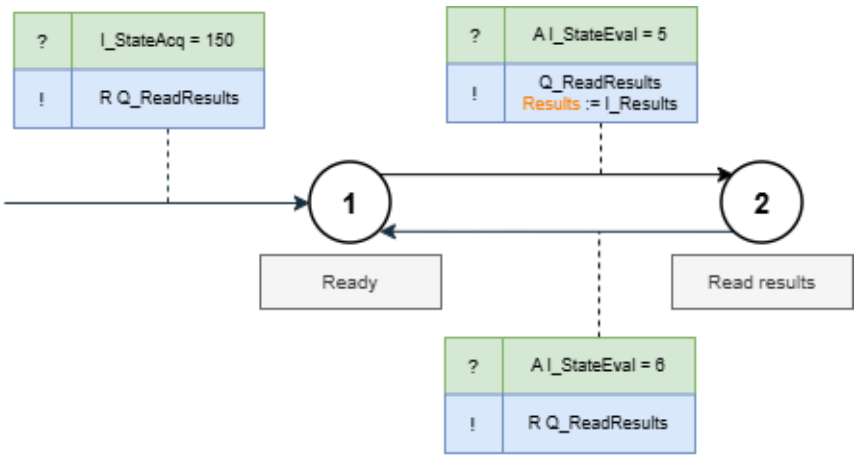


Figure 16: Signal Sequence of the Evaluation State Machine from the Perspective of the Communication Partner

Communication partner

The state transition of the MEAutomationCore (I_StateEval = 5) is the transition condition for the communication partner. In the process, it transfers the measurement results from Q_ReadResults to the automation environment (Results) and sets Q_ReadResults as a signal that the measurement results have been fetched.

With I_StateEval = 6, the communication partner changes back to state 1 and resets its Q_ReadResults again.

MEAutomationCore

After Acquisition State 4, the data is transferred to the Evaluation state machine for evaluation there.
When the data is transferred, the Evaluation state machine changes from state 1 to state 4 and resets Q_Results in the process. As soon as the evaluation is complete, the Evaluation state machine changes to state 5 and writes its results to Q_Results.

The signal I_ReadResults informs the MEAutomationCore that the measurement results are read, and it changes to state 6.

The MEAutomationCore knows from the reset I_ReadResults that the process has been completed and can change back to its state 1.

Measurement Sequence in Continuous Operation

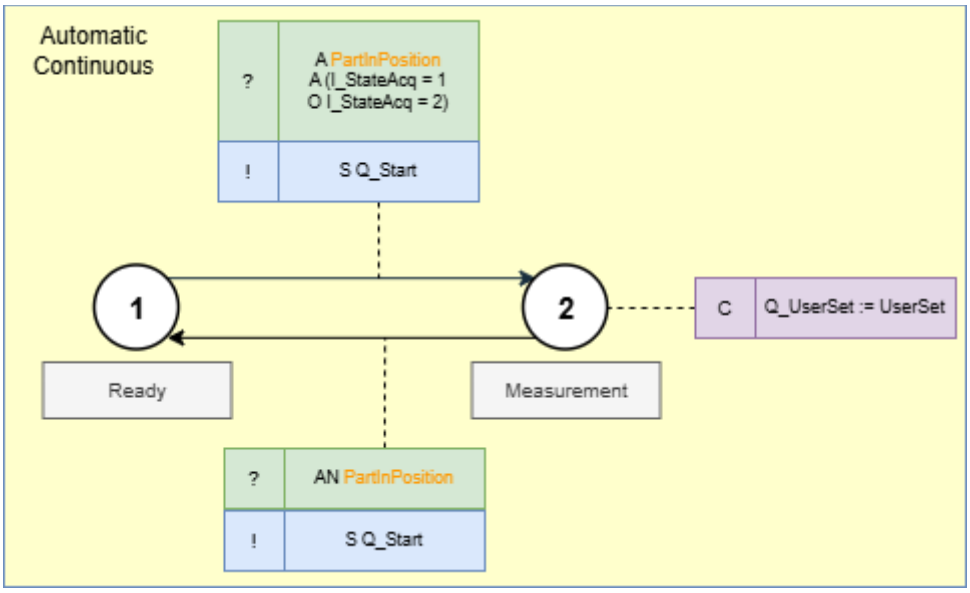


Figure 17: Measurement in the Continuous Case from the Perspective of the Communication Partner

Communication partner	MEAutomationCore
In Ready State 1, if the request from the automation environment for a measurement is set by PartInPosition and the MEAutomationCore is also in Ready State (I_StateAcq = 1), the communication partner sets Q_Start and changes to state 2.	
	When the start signal of the communication partner is read and the currently output UserSet corresponds to the requested UserSet, continuous measurement and processing are started and the state changes to state 2.
The measurement results (I_Results) can be read continuously and transferred to the automation environment (Results). When PartInPosition is reset, the communication partner resets Q_Start and changes to state 1.	
	As soon as the I_Start signal is no longer present, the measurement is aborted and the state changes back to state 1.

3. Appendix C Server State Machines (Sensor Side)

To make it possible in the discrete case to start the next data acquisition while the previous data evaluation is still being evaluated, there are two state machines (Acquisition and Evaluation).

Discrete Case:

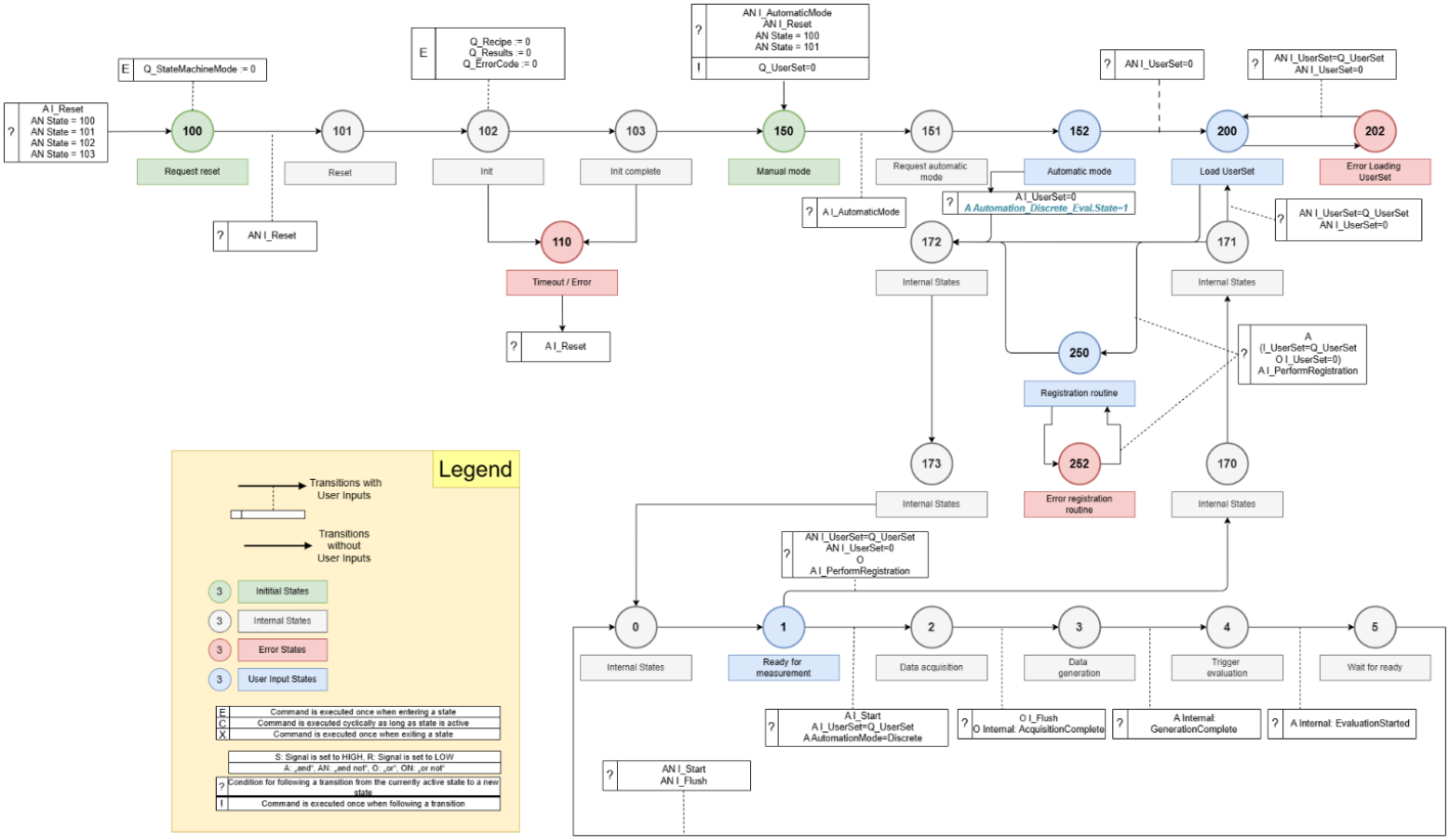


Figure 18: MEAutomationCore State Machine for discrete acquisition

The diagram above shows the "Acquisition" state machine (data acquisition), which runs on the Micro-Epsilon sensor in the discrete case.

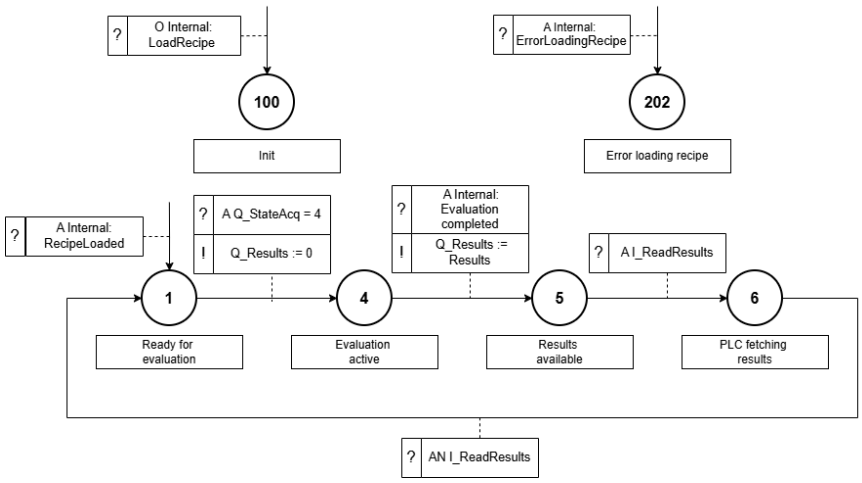


Figure 19: MEAutomationCore discrete evaluation

The diagram above shows the "Evaluation" state machine (data evaluation), which runs on the Micro-Epsilon sensor in the discrete case.

Continuous Case

In the continuous case as well, the Acquisition and Evaluation state machines run in parallel.

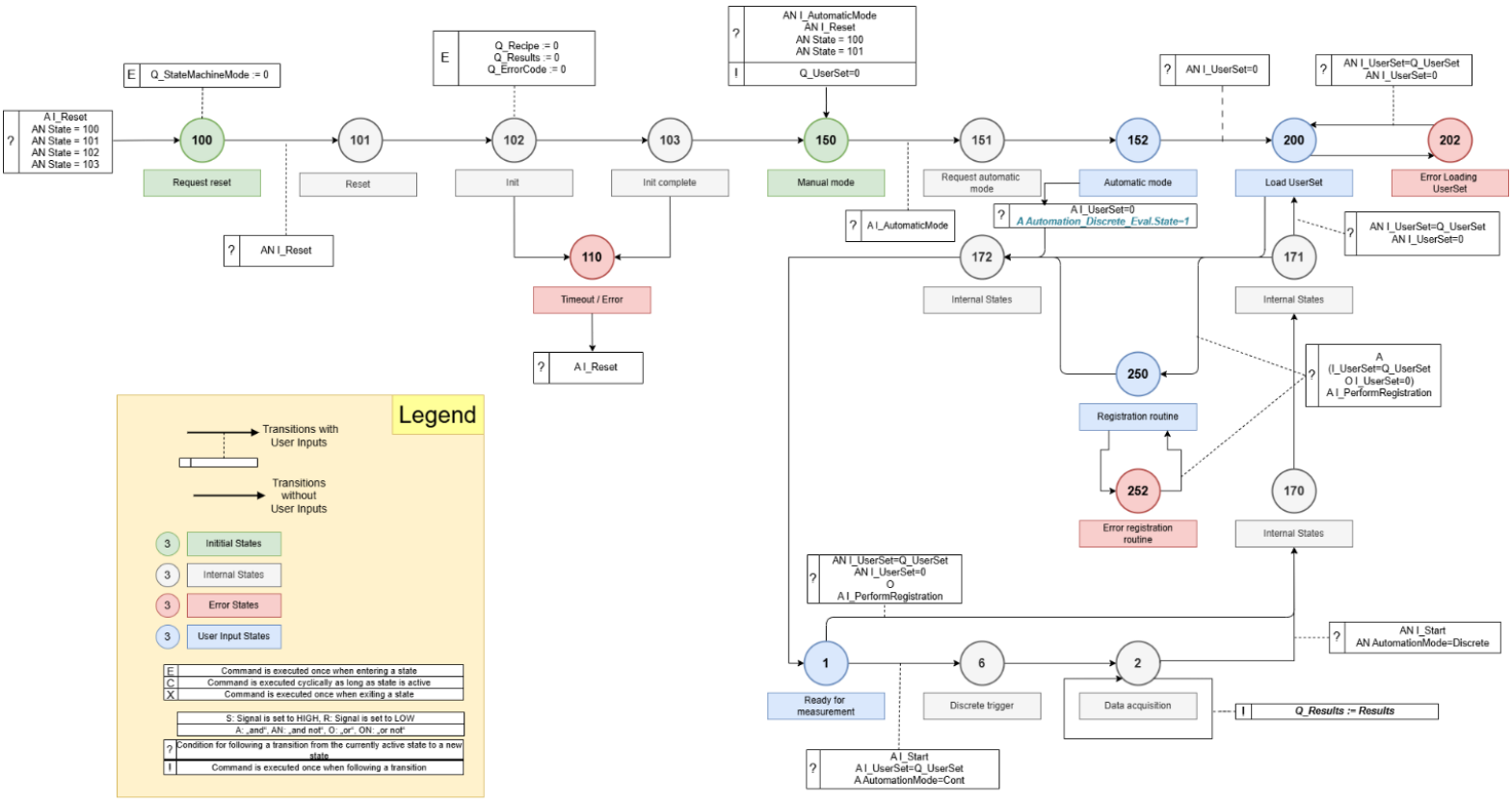


Figure 20: MEAutomationCore continuous acquisition

The diagram above shows the "Acquisition" state machine (data acquisition), which runs on the Micro-Epsilon sensor in the continuous case.

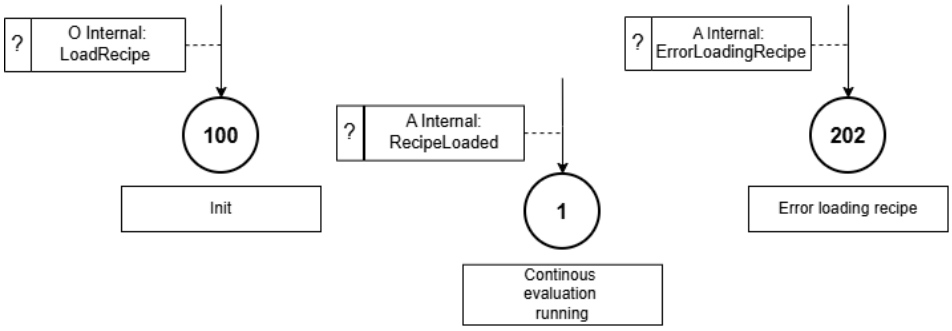


Figure 21: MEAutomationCore continuous evaluation

The diagram above shows the "Evaluation" state machine (data evaluation), which runs on the Micro-Epsilon sensor in the continuous case. In this special case, only unconditional state transitions are found.

MICRO-EPSILON MESSTECHNIK GmbH & Co. KG
Königbacher Str. 15 · 94496 Ortenburg / Deutschland
Tel. +49 (0) 8542 / 168-0 · Fax +49 (0) 8542 / 168-90
info@micro-epsilon.de · www.micro-epsilon.de
Your local contact: www.micro-epsilon.com/contact/worldwide/

X9751515-A012066SDR