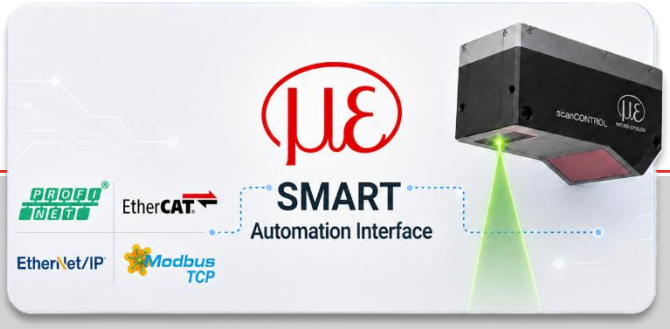
MICRO-EPSILON

Bedienungsanleitung



Micro-Epsilon SMART Automation Interface 2.3

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

D-94496 Ortenburg / Deutschland

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

SMART Automation Interface // Dokumentation

Inhalt

I. Allgemeine Beschreibung 4
II. Einrichtung in 3DInspect 5
1. Vorbereitung der Messaufgabe 5
2. Schnittstelle auswählen und Parametersätze hinterlegen 5
3. Einrichtbetrieb – Messbetrieb 6
III. Realisierte Funktionen und Signale 7
1. Steuer- und Vorgabesignale 7
2. Rückmeldesignale 7
3. Datenbereiche 8
IV. Einbindung über Modbus TCP 9
Registerbelegungen/Interface-Signale 10
V. Einbindung über Feldbusse 11
1. Interface-Signale 12
2. PROFINET 13
3. EtherCAT 14
4. EtherNet/IP 15
VI. Grundablauf bis zur ersten Messung 16
VII. Anhang 17
1. Anhang A: Fehler-Codierung: 17
2. Anhang B: Beispielhafte State Machine des Kommunikationspartners 18
3. Anhang C Server State Machines (sensor-seitig) 26

I. Allgemeine Beschreibung

Der **MEAutomationCore** ermöglicht die automatisierte Ansteuerung von MICRO-EPSILON SMART-Sensoren über industrielle Kommunikationsschnittstellen. Über das Interface können Messungen ausgelöst, Messwerte übertragen, Parametersätze geladen, Resets durchgeführt und weitere Steuerfunktionen genutzt werden.

Der **MEAutomationCore** beschreibt die sensorseitige Ablaufsteuerung. Dazu gehören insbesondere die Acquisition- und Evaluation-State-Machines.

Das **SMART Automation Interface** beschreibt die standardisierte Schnittstelle zwischen MEAutomationCore und dem externen Kommunikationspartner, zum Beispiel SPS, PC, Modbus-Client oder Feldbus-Controller.

Die Anbindung kann je nach Systemumgebung über folgende Schnittstellen erfolgen:

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

Diese Dokumentation richtet sich an **SPS-Programmierer, Integratoren und Softwareentwickler**, die einen MICRO-EPSILON SMART-Sensor in eine Automatisierungsumgebung einbinden möchten.

Das Interface basiert auf einem definierten Signalablauf. Dabei durchläuft der **MEAutomationCore** verschiedene Zustände, die als **State Machine** abgebildet sind. Die Zustände müssen in einer festgelegten Reihenfolge durchlaufen werden. Der Kommunikationspartner, zum Beispiel eine SPS oder ein Modbus-Client, muss diese Zustände auswerten und die entsprechenden Steuersignale setzen.

Dadurch entsteht ein robuster und nachvollziehbarer Kommunikationsablauf zwischen MEAutomationCore und Steuerung. Für eine stabile Anbindung wird empfohlen, auch auf der Client-Seite eine eigene State Machine umzusetzen.

Die notwendigen Einstellungen werden über **3DInspect** vorgenommen. Dazu gehören insbesondere die Vorbereitung der Parametersätze, die Auswahl des Betriebsmodus sowie schnittstellenspezifische Einstellungen wie zum Beispiel die Endianness

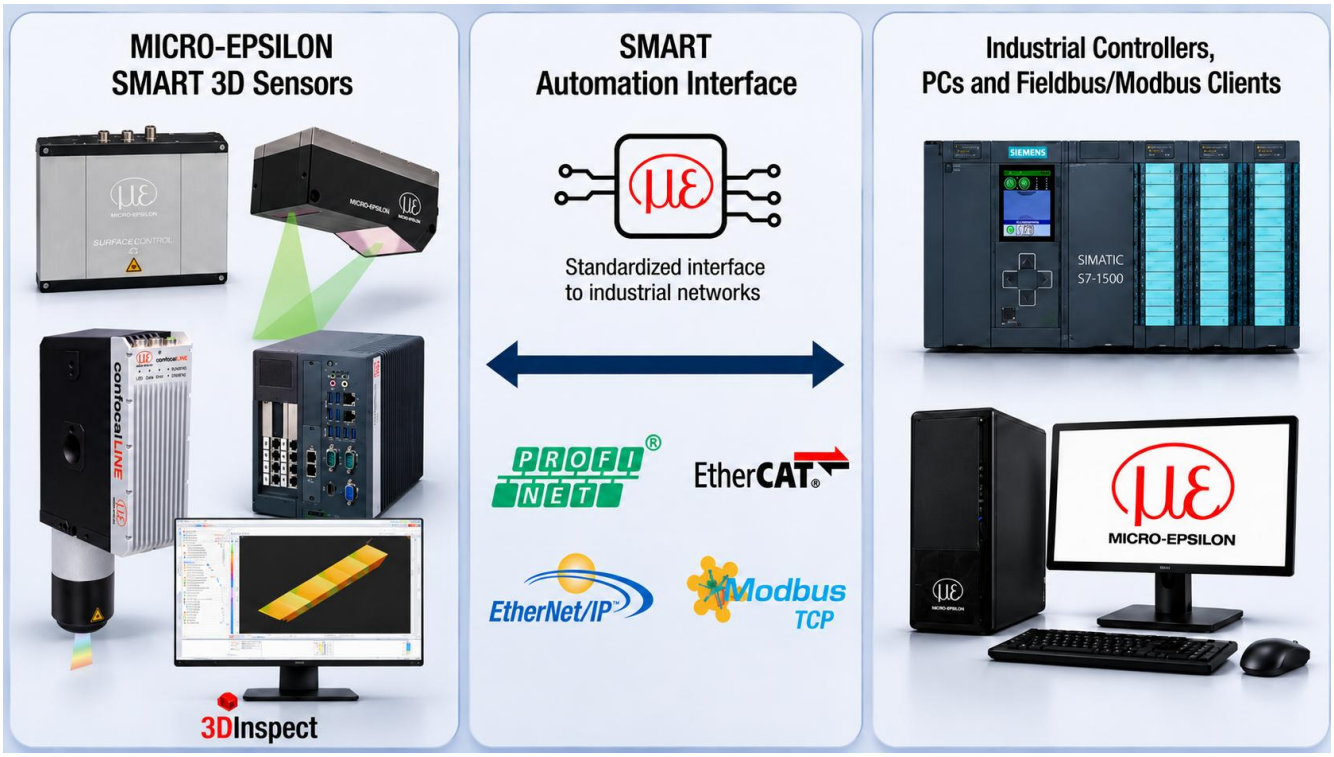


Abbildung 1: SMART Automation Interface

II. Einrichtung in 3DInspect

Vor der Nutzung des MEAutomationCore muss der Sensor in 3DInspect für den automatisierten Betrieb vorbereitet werden. Die genaue Konfiguration hängt von der Applikation, dem Sensortyp und der verwendeten Kommunikationsschnittstelle ab.

1. Vorbereitung der Messaufgabe

Vor dem automatisierten Betrieb muss die Messaufgabe in 3DInspect vollständig eingerichtet und getestet werden.

Dazu gehören insbesondere:

- Aufnahme- und Auswerteparameter,
- benötigte Ergebniswerte,
- vorbereitete Parametersätze, die zunächst auf dem PC gespeichert werden
- erforderliche Schnittstelleneinstellungen.

Details zur Einrichtung und Parametrierung der Messaufgabe in 3DInspect sind der jeweiligen 3DInspect-Dokumentation zu entnehmen.

2. Schnittstelle auswählen und Parametersätze hinterlegen

Für den automatisierten Betrieb müssen in 3DInspect zunächst die verwendete Schnittstelle und die zugehörigen Einstellungen ausgewählt werden. Anschließend werden die benötigten Parametersätze hinterlegt. Der Client gibt später die Nummer des zu ladenden UserSets / Rezepts an den MEAutomationCore vor.

Die Einstellungen erfolgen im Bereich „Ergebnisse ausgeben“ über den Reiter „Eingabe/Ausgabe“.

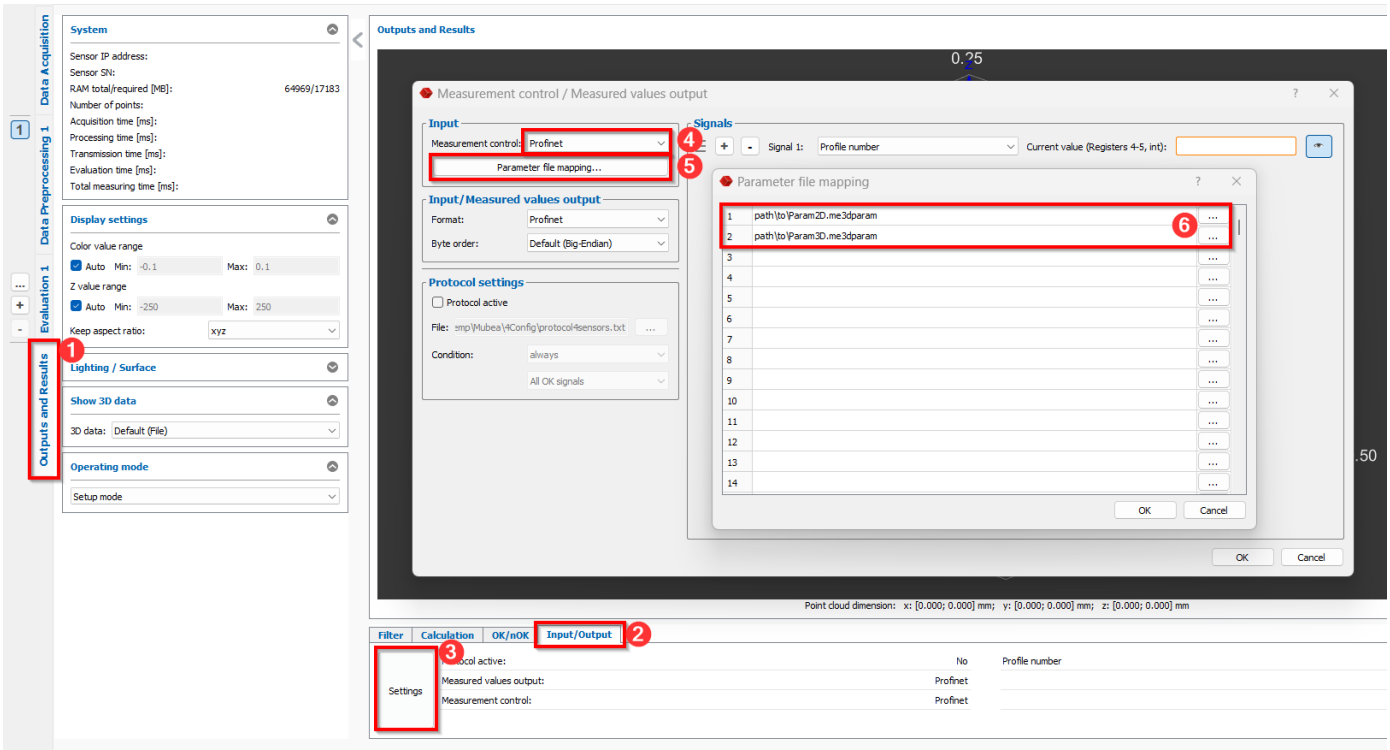


Abbildung 2: 3DInspect

Vorgehensweise:

- 1. Den Bereich „Ergebnisse ausgeben“ öffnen.
- 2. Den Reiter „Eingabe/Ausgabe“ auswählen.
- 3. Die Einstellungen öffnen.
- 4. Im Bereich „Eingabe/Ausgabe Messwerte“ das gewünschte Format auswählen, zum Beispiel Modbus, EtherCAT, EtherNet/IP oder PROFINET.
- 4.1 Im Bereich „Eingabe“ unter „Steuerung Messung“ die verwendete Netzwerkschnittstelle auswählen. Die Option „Sensor HW-Trigger“ wird für die Ansteuerung über den MEAutomationCore nicht unterstützt.
- 4.2 Bei Feldbusanbindungen die passende Byte-Reihenfolge einstellen:
    - PROFINET: Default Big-Endian
    - EtherCAT: Default Little-Endian
    - EtherNet/IP: Default Little-Endian
- 4.3 Bei Modbus den Modbus-Port prüfen. Standardmäßig wird Port 502 verwendet.
- 5. Die Schaltfläche „Zuordnung Parameterdateien…“ auswählen.
- 6. Die gewünschten Parameterdateien den jeweiligen Nummern zuordnen.
- 7. Die Zuordnung mit „OK“ bestätigen.

Die Nummer der jeweiligen Zeile entspricht der Nummer, die später vom Client vorgegeben wird. Dadurch kann der MEAutomationCore den zugehörigen auf dem PC gespeicherten laden.

Die Byte-Reihenfolge muss mit der Einstellung auf Client-Seite übereinstimmen. Ist sie falsch eingestellt, werden Statuswerte, Fehlercodes oder Messergebnisse falsch interpretiert.

3. Einrichtbetrieb – Messbetrieb

Für den Betrieb des Sensors wird zwischen Einrichtbetrieb und Messbetrieb unterschieden.

Im Einrichtbetrieb wird der Sensor in 3DInspect eingerichtet und getestet. In diesem Zustand können Parameter angepasst, Messaufgaben geprüft und Parametersätze vorbereitet werden.

Für den automatisierten Betrieb muss anschließend in 3DInspect in den Messbetrieb geschaltet werden. Erst im Messbetrieb kann der Sensor über den MEAutomationCore durch den verbundenen Client gesteuert werden.

Der Client kann dann zum Beispiel den Automatikmodus anfordern, Parametersätze laden, Messungen starten, Ergebnisse auslesen oder Fehler quittieren.

Die Steuerung über den MEAutomationCore ist nur möglich, wenn der Sensor in 3DInspect eingerichtet wurde, kein aktiver Fehler anliegt und die Feldbus- oder Modbus Kommunikation zum Sensor besteht.

III. Realisierte Funktionen und Signale

Die folgenden Funktionen und Signale werden über das SMART Automation Interface zur Ansteuerung des MEAutomationCore bereitgestellt. Sie können je nach Kommunikationsart über Modbus TCP, PROFINET, EtherCAT oder EtherNet/IP abgebildet werden.

Der Begriff Client beschreibt den Kommunikationspartner des MEAutomationCore, zum Beispiel eine SPS, einen PC, einen Modbus-Client oder einen Feldbus-Controller.

1. Steuer- und Vorgabesignale

Monitoring (bMonitoring):

Aktiviert die Monitoring-Funktion. Der SMART-Sensor kann so Messdaten und Messwerte an 3DInspect senden zur Visualisierung.

Parametersatz auswählen (nUserSet):

Gibt die Nummer des zu ladenden UserSets / Rezepts vor. Die Nummer muss der in 3DInspect hinterlegten Parameterdatei entsprechen. Es können bis zu 255 UserSets / Rezepts geladen werden.

Job Sequence Number (sJSN):

Übergibt eine Job Sequence Number mit bis zu 40 Zeichen vom Client an den MEAutomationCore. Dadurch können Messdaten und Messergebnisse Produktionsdaten zugeordnet werden.

Emitter aus (EmitterOff):

Schaltet die Lichtquelle des Sensors aus.

Fehler quittieren (ResetError):

Quittiert einen anstehenden Fehler über eine positive Flanke. Die Fehlerbehandlung und Speicherung des Fehlers muss durch den Anwender umgesetzt werden.

Reset (bReset):

Startet die Reset-Routine. Dabei werden die Schnittstellensignale zurückgesetzt und die State Machine in einen definierten Ausgangszustand überführt.

Automatikmodus anfordern (bAutomaticMode):

Fordert den Automatikmodus an. Dies ist möglich, wenn sich 3DInspect nicht im Einrichtbetrieb befindet.

Automatische Registrierung (bAutomaticRegistration):

Startet bei Sensoren mit Registrierroutine die automatische Registrierung.

Messung starten (bInPosition_Start):

Löst im diskreten Betrieb eine Messung aus. Das Signal sollte gesetzt werden, wenn sich das Bauteil in Messposition befindet. Im diskreten Betrieb wird das Startsignal nach der Übernahme automatisch zurückgesetzt.

Im kontinuierlichen Betrieb startet der Sensor die Messzyklen selbstständig. so lange das Startsignal anliegt

Flush (bFlush):

Beendet bei Linien-Sensoren eine laufende 3D-Messung per Flush, auch wenn die Profile für eine 3D Punktewolke noch nicht vollständig erfasst wurden.

2. Rückmeldesignale

Live-Signal (Live):

Das Live-Signal wird mit einer Frequenz von 2 Hz getoggelt und dient zur Überwachung der Kommunikation zwischen MEAutomationCore und Client.

Bauteil entladen (bUnloadPart):

Wird auf TRUE gesetzt, sobald die Belichtung beziehungsweise Datenaufnahme der Messung abgeschlossen ist. Das Bauteil kann danach weitertransportiert werden, während die Auswertung der aufgenommenen Messdaten weiterläuft.

Fehlercode (ErrorCode):

Gibt den aktuell anstehenden Fehlercode aus.

Status Acquisition (InStateAcq):

Zeigt den aktuellen Zustand der Acquisition State Machine an.

Status Evaluation (InStateEval):

Zeigt den aktuellen Zustand der Evaluation State Machine an.

Aktives UserSet / Rezept (UserSet):

Gibt den aktuell auf dem Sensor geladenen UserSet / Rezept aus.

Validierte Ergebnisse (Results):

Stellt die validierten Messergebnisse zur weiteren Verarbeitung bereit.

3. Datenbereiche

Basis-Eingangsdaten (INBasis):

Enthält grundlegende Rückmeldungen des MEAutomationCore, zum Beispiel Statusinformationen, Live-Signal und Fehlerzustände.

Smart-Eingangsdaten (INSmart):

Enthält erweiterte Rückmeldungen des MEAutomationCore, zum Beispiel Zustände der State Machines.

Unvalidierte Ergebnisse (INResults):

Enthält die vom MEAutomationCore bereitgestellten Ergebnisdaten.

Basis-Ausgangsdaten (OUTBasis):

Enthält grundlegende Steuerbefehle, die vom Client an den MEAutomationCore übertragen werden.

Smart-Ausgangsdaten (OUTSmart):

Enthält erweiterte Steuerbefehle, zum Beispiel Automatikmodus, Startsignal, Flush und Ergebnisquittierung.

Job Sequence Number (OUTJSN):

Überträgt die Job Sequence Number zur Zuordnung der Messwerte zu den Produktionsdaten.

IV. Einbindung über Modbus TCP

Um das SMART Automation Interface über Modbus TCP nutzen zu können, muss zunächst eine TCP-Verbindung zu Port 502 des verwendeten Sensors aufgebaut werden.

Grundlegende Modbus-Befehle

Für die Kommunikation selber sind zwei Modbus-Befehle zu verwenden (Indexbase 1):

- **03 - Read Input Registers**
  Startregister: 0
  Anzahl Register: 128
- **16 - Write Multiple Holding Registers**
  Startregister: 0
  Anzahl Register: 24

Zyklische Datenerfassung

Für den laufenden Betrieb wird empfohlen, den Befehl Read Input Registers zyklisch auszuführen, um Änderungen im Automation Core des Sensors kontinuierlich zu erfassen.

Die Abfrage muss regelmäßig erfolgen. Eine Frequenz von 1 Hz stellt dabei das absolute Minimum dar. Bei einer niedrigeren Abfragefrequenz geht der MEAutomationCore von einer gestörten Kommunikation aus und schließt die Verbindung entsprechend.

Für einen stabilen und performanten Betrieb werden folgende Richtwerte empfohlen:

- 3D-Messung: typischerweise 20-50 Hz
- 2D-Profilmessung: deutlich höhere Messfrequenzen; hier kann eine schnellere Abfrage oder alternativ eine andere Übertragungsmethode für die Messwerte erforderlich sein

Schreibzugriffe

Der Befehl Write Multiple Holding Registers wird nur bei Bedarf ausgeführt, beispielsweise zur Parametrierung oder Steuerung des Sensors.

Beispielimplementierung

Auf der Micro-Epsilon-Website steht mit dem „Micro-Epsilon Modbus-Tool“ eine Beispielimplementierung für die Modbus/TCP-Kommunikation in Python zur Verfügung. Das Tool ermöglicht durch die anschauliche Visualisierung des Zustandsautomaten sowie der übertragenen Parameter einen schnellen und intuitiven Einstieg in das Verständnis der Modbus-Kommunikation.

Nach dem Start des Tools kann im Reiter „Terminalbeispiel“ über die Schaltfläche „Quellcode speichern“ ein ZIP-komprimierter Ordner mit dem Python-Quellcode heruntergeladen werden. Dieser bildet die grundlegende Struktur der Kommunikation mit dem MEAutomationCore über Modbus ab. Zusätzlich enthält das Paket Beispielimplementierungen, die zeigen, wie sich der MEAutomationCore beispielsweise über die Kommandozeile steuern lässt oder wie eine erste Messung durchgeführt werden kann.

Damit eignet sich das Modbus-Tool zugleich als kompaktes Einstiegstutorial in die Python-Programmierung im Kontext der Modbus-Kommunikation.

Schematischer Ablauf

Es wird empfohlen, folgenden Ablauf im Client-Programm zu befolgen, um eine reibungslose Kommunikation mit dem MEAutomationCore zu ermöglichen:

1. Einlesen der Modbus-Antwort auf das zuvor gesendete Kommando.
2. Durchlaufen der State Machine *Acquisition* auf Client-Seite und Setzen der erforderlichen Interface-Signale.
   → Falls ein Schreibzugriff auf Holding Register erforderlich ist, diesen Bedarf vormerken.

3. Durchlaufen der State Machine *Evaluation* auf Client-Seite und Setzen der erforderlichen Interface-Signale.
   → Falls ein Schreibzugriff auf Holding Register erforderlich ist, ebenfalls vormerken.

4. Prüfung des vorgemerkten Bedarfs:
   → Falls ein Schreibzugriff notwendig ist, Durchführung von *Write Holding Registers*.
   → Andernfalls erneute Durchführung von *Read Input Registers*.

Registerbelegungen/Interface-Signale

Nachfolgende Tabelle zeigt die Signale des SMART Automation Interface, die zur Ansteuerung des MEAutomationCore notwendig sind.. Aus der Sicht des Kommunikationspartners/Clients sind die Holding Register die Ausgangsdaten, die zum Sensor geschickt werden. Die Input Register repräsentieren die Eingangs-Signale, die empfangen werden.

		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	

Abbildung 3: MEAutomationInterface Signale

V. Einbindung über Feldbusse

Das SMART Automation Interface kann über PROFINET, EtherCAT oder EtherNet/IP in eine Feldbusumgebung eingebunden werden. Der Datenaustausch erfolgt dabei zwischen Client und MEAutomationCore über den jeweiligen Feldbus.

Für jeden Feldbus stehen die notwendigen Dateien zur Einbindung zur Verfügung. Dazu gehören die jeweilige Gerätebeschreibungsdatei, ein Beispielprogramm mit dem **SMART Automation Interface** sowie die zugehörige **SMART Automation Interface Library**.

Für die Einbindung wird die passende Gerätebeschreibungsdatei im jeweiligen Engineering-Tool benötigt:

- PROFINET: GSDML-Datei, z. B. für Siemens TIA Portal
- EtherCAT: ESI-Datei, z. B. für TwinCAT
- EtherNet/IP: EDS-Datei, z. B. für Allen-Bradley

Die verwendete Schnittstelle und die passende Byte-Reihenfolge (Endianness) müssen zuvor in 3DInspect eingestellt werden. Die Vorgehensweise ist in Kapitel II, Abschnitt 2 „Schnittstelle auswählen und Parametersätze hinterlegen“ beschrieben.

Konfiguration der Ergebnisdaten

Standardmäßig ist der Ergebnisbereich für **64 Ergebnisse** konfiguriert. Bei der Konfiguration kann die Größe des Datenbereichs für die übertragenen Ergebnisdaten variabel festgelegt werden. Sind mehrere Parametersätze hinterlegt, muss die Datenstruktur immer für den Parametersatz mit dem **größten Ergebnisumfang** ausgelegt werden. Dadurch ist sichergestellt, dass die Ergebnisdaten aller hinterlegten Parametersätze vollständig übertragen werden können.

Hinweis: Bei der Auslegung des Ergebnisbereichs ist zu berücksichtigen, dass eine **JSN 10 Results-Blöcke benötigt**.

1. Interface-Signale

Die Interface-Signale des SMART Automation Interface sind für alle Feldbusse gleich aufgebaut. Die Richtung der Daten wird aus Sicht des Clients beziehungsweise der Steuerung beschrieben.

Die Ausgangsdaten werden vom Client an den MEAutomationCore übertragen. Die Eingangsdaten werden vom MEAutomationCore an den Client übertragen.

Die genaue Aufteilung der Datenbereiche und die Belegung der einzelnen Signale sind in der nachfolgenden Abbildung dargestellt.

	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	

Abbildung 4: MEAutomationInterface Signale

2. PROFINET

GSDML-Datei bereitstellen und importieren

Öffnen Sie im TIA Portal den Dialog Extras → Gerätebeschreibungsdateien (GSD) verwalten. Wählen Sie den Ordner aus, in dem die GSDML-Datei abgelegt ist, markieren Sie die Datei und klicken Sie auf „Installieren“.

Hinweis: Eine detaillierte Beschreibung zur Installation von GSD-Dateien ist in der Siemens-Dokumentation beschrieben:

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

Hardware aus dem Katalog hinzufügen

Öffnen Sie im Projekt unter Geräte & Netze den Hardware-Katalog.Pfad:

Weitere Feldgeräte → PROFINET IO → Sensors → MICRO-EPSILON MESSTECHNIK → [Produktfamilie] → [Sensor]

Fügen Sie das Gerät per Drag & Drop in die Netzansicht ein und konfigurieren Sie es.

Parameter setzen und Verbindung prüfen

Die IP-Adresse des Sensors muss im gleichen Subnetz wie die Steuerung liegen. Zusätzlich muss ein eindeutiger Gerätename vergeben werden. Sonderzeichen und Leerzeichen sollten dabei vermieden werden.

Anschließend wird das Projekt übersetzt und in die Steuerung geladen. Nach dem Online-Verbindungsaufbau muss das Gerät fehlerfrei angezeigt werden. Die zyklische Kommunikation kann über das toggelnde Live-Bit geprüft werden.

Ergebnisbereich anpassen

Stellen Sie im PLC-Datentyp UDT_INResults die Arraygröße der Variable Results so ein, dass sie der in der Hardwarekonfiguration festgelegten Größe des Ergebnisbereichs entspricht.

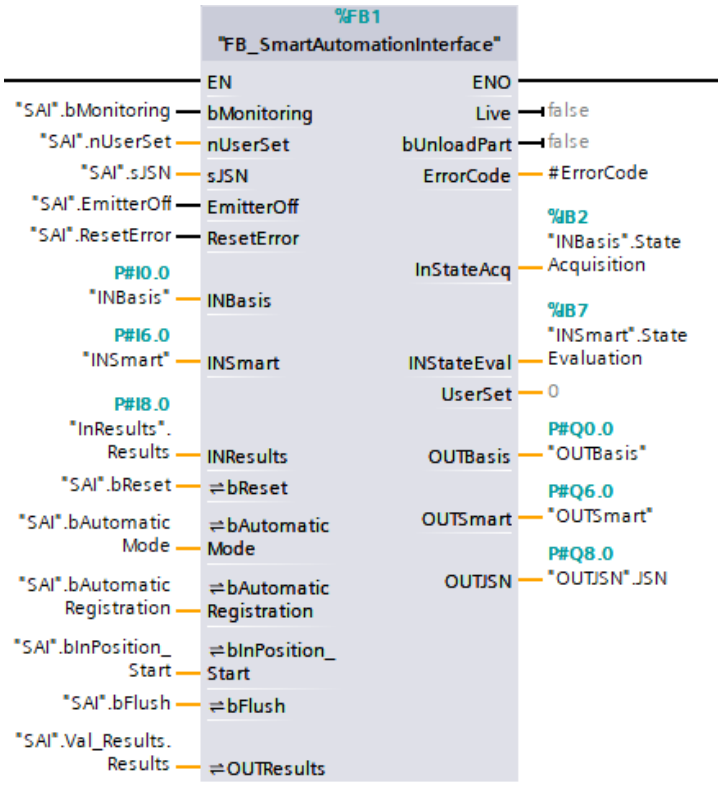


Abbildung 5:Einbindung PROFINET

3. EtherCAT

ESI-Datei bereitstellen und importieren

Kopieren Sie die ESI-Datei in das EtherCAT-Verzeichnis von TwinCAT. Standardpfad:

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

Starten Sie anschließend TwinCAT 3, damit die Gerätebeschreibung eingelesen wird.

Hardware aus dem Katalog hinzufügen

Führen Sie im Projektbaum unter I/O → Devices den Befehl Scan aus.

Das gefundene EtherCAT-Gerät übernehmen und die anschließende Suche nach Boxen mit „Ja“ bestätigen. Das Gerät wird danach automatisch in die EtherCAT-Topologie eingefügt und kann konfiguriert werden.

Variablen zuordnen

Die Ein- und Ausgangsvariablen des SPS-Projekts müssen den Interface-Signalen des SMART Automation Interface zugeordnet werden.

Hierzu werden die betreffenden Variablen markiert und über die Mehrfachverknüpfung mit den entsprechenden Hardwarekanälen verbunden.

Dabei ist auf die korrekte Zuordnung und Reihenfolge der Ein- und Ausgangsdaten zu achten.

Verbindung prüfen

Aktivieren Sie die Konfiguration und bauen Sie die Online-Verbindung zum EtherCAT-System auf.

Zur Prüfung der Kommunikation kann das Live-Bit verwendet werden. Wenn das Live-Bit toggelt, werden die Daten zwischen Sensor und Client zyklisch übertragen.

Ergebnisbereich anpassen

Stellen Sie in der globalen Variablenliste GVL_SAI den Wert der Konstante cResultCount entsprechend der in der Hardwarekonfiguration festgelegten Größe des Ergebnisbereichs ein.

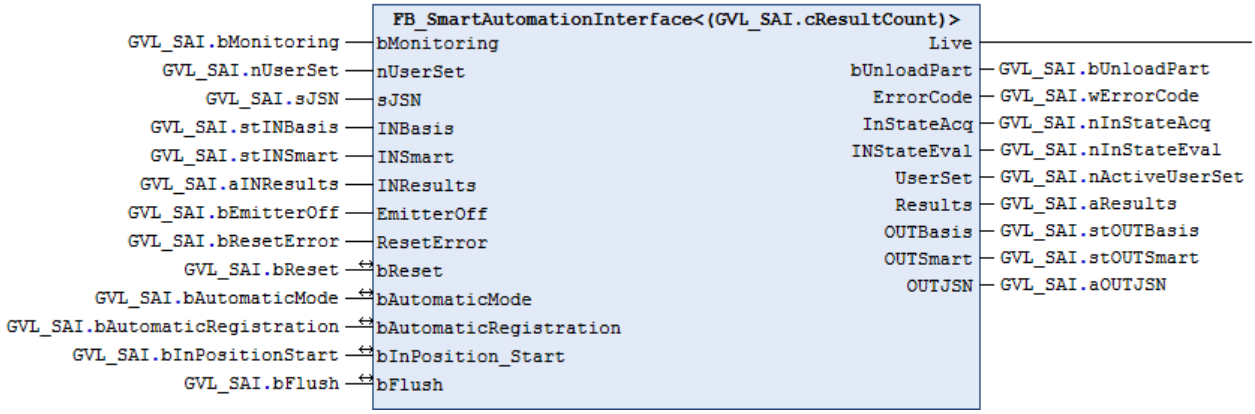


Abbildung 6: Einbindung EtherCAT

4. EtherNet/IP

EDS-Datei bereitstellen und importieren

Installieren Sie die zum verwendeten Gerät gehörende EDS-Datei in der Rockwell-Engineering-Umgebung, zum Beispiel über das EDS Hardware Installation Tool. Nach erfolgreicher Installation steht das Gerät im Hardware-Katalog zur Verfügung.

IP-Adresse vergeben

Die IP-Adresse des EtherNet/IP-Teilnehmers muss im gleichen Subnetz wie die Steuerung liegen. Bei der Erstinbetriebnahme kann die Adresse zum Beispiel mit dem Rockwell BOOTP/DHCP-Tool über die MAC-Adresse des Geräts vergeben werden.

Nach der Adressvergabe muss BOOTP/DHCP deaktiviert beziehungsweise der Adressmodus auf eine statische IP-Adresse umgestellt werden. Anschließend sollte das Gerät neu gestartet und geprüft werden, ob die IP-Adresse erhalten bleibt.

Hardware aus dem Katalog hinzufügen

Öffnen Sie im Studio-5000-Projekt die I/O-Konfiguration des verwendeten Ethernet-Scanners beziehungsweise Ethernet-Moduls. Fügen Sie das Gerät aus dem Hardware-Katalog hinzu und tragen Sie die zuvor vergebene IP-Adresse ein.

Die im Projekt eingetragene IP-Adresse muss mit der tatsächlich am EtherNet/IP-Teilnehmer eingestellten IP-Adresse übereinstimmen.

Verbindung prüfen

Laden Sie das Projekt in die Steuerung und bauen Sie die Online-Verbindung auf. Das Gerät muss fehlerfrei angezeigt werden. Die zyklische Kommunikation mit dem **MEAutomationCore** kann über das Live-Bit geprüft werden. Wenn das Live-Bit toggelt, werden die Daten zwischen **MEAutomationCore** und Steuerung zyklisch übertragen.

Ergebnisbereich anpassen

Stellen Sie im benutzerdefinierten Datentyp UDT_INResults die Arraygröße der Variable Results so ein, dass sie der in der Hardwarekonfiguration festgelegten Größe des Ergebnisbereichs entspricht.

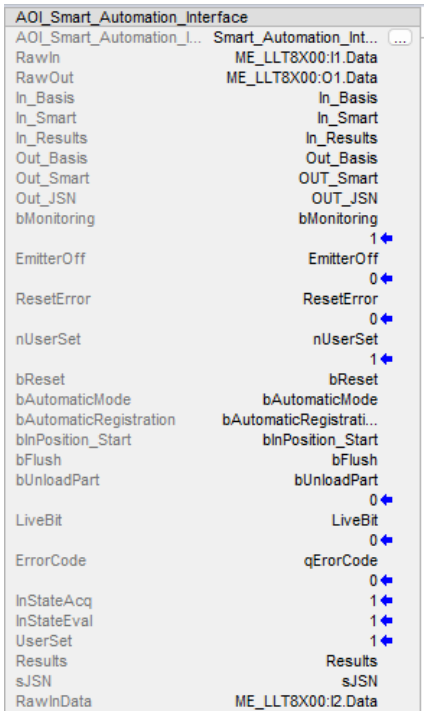


Abbildung 7: Einbindung EtherNet/IP

VI. Grundablauf bis zur ersten Messung

Voraussetzung:

Die Kommunikation zum MEAutomationCore ist aufgebaut, es liegt kein aktiver Fehler an und das Live-Bit toggelt.

1. Grundzustand prüfen

Warten Sie, bis der MEAutomationCore den Acquisition State 150 erreicht hat. In diesem Zustand befindet sich das System im Manual Mode.

2. UserSet / Rezept vorgeben

Geben Sie das gewünschte UserSet / Rezept vor. Der UserSet / Rezept wird nach dem Aktivieren des AutomaticMode vom MEAutomationCore geladen.

3. Automatikmodus aktivieren

Aktivieren Sie den AutomaticMode.

4. Ladevorgang abwarten

Während des Ladens wechselt der MEAutomationCore in den Acquisition State 200, um das UserSet / Rezept zu laden.

5. Messbereitschaft prüfen

Nach erfolgreichem Laden wechselt der MEAutomationCore in den Acquisition State 1 (Ready). In diesem Zustand ist das System messbereit.

6. Messung starten

Sobald I_StateAcq = 1 anliegt, kann die Messung über das Startsignal bInPosition_Start ausgelöst werden.

VII. Anhang

1. Anhang A: Fehler-Codierung ErrorCode-Feld:

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.

Abbildung 8: Fehler-Codierung

2. Anhang B: Beispielhafte State Machine des Kommunikationspartners

Es wird empfohlen, die Ablaufsteuerung auf Seite des Kommunikationspartners ebenfalls als State Machine umzusetzen. Der Kommunikationspartner kann zum Beispiel eine SPS, ein Modbus-Master, ein PC-Programm oder ein anderer Client sein.

Die folgenden Diagramme zeigen eine beispielhafte Umsetzung aus Sicht des Kommunikationspartners. Sie dienen als Grundlage für eigene Implementierungen und müssen je nach Applikation angepasst werden.

Darstellung der Signalabläufe

Die Diagramme zeigen, welche Zustände durchlaufen werden und welche Signale dabei gesetzt, zurückgesetzt oder ausgewertet werden.

Die **schwarz** dargestellten Signale werden direkt zwischen Kommunikationspartner und MEAutomationCore ausgetauscht.

Dabei gilt:

- I_: Eingangssignal des Kommunikationspartners, kommt vom MEAutomationCore

  Beispiel: I_StateAcq, I_StateEval, I_Results

- Q_: Ausgangssignal des Kommunikationspartners, geht an den MEAutomationCore

  Beispiel: Q_Reset, Q_AutomaticMode, Q_UserSet, Q_Start

Die orange dargestellten Signale gehören zur übergeordneten Automatisierungsumgebung, zum Beispiel zu Ein- und Ausgängen eines SPS-Bausteins. Beispiele sind **Reset**, **AutomaticMode**, **UserSet**, **PartInPosition** oder **UnloadPart**.

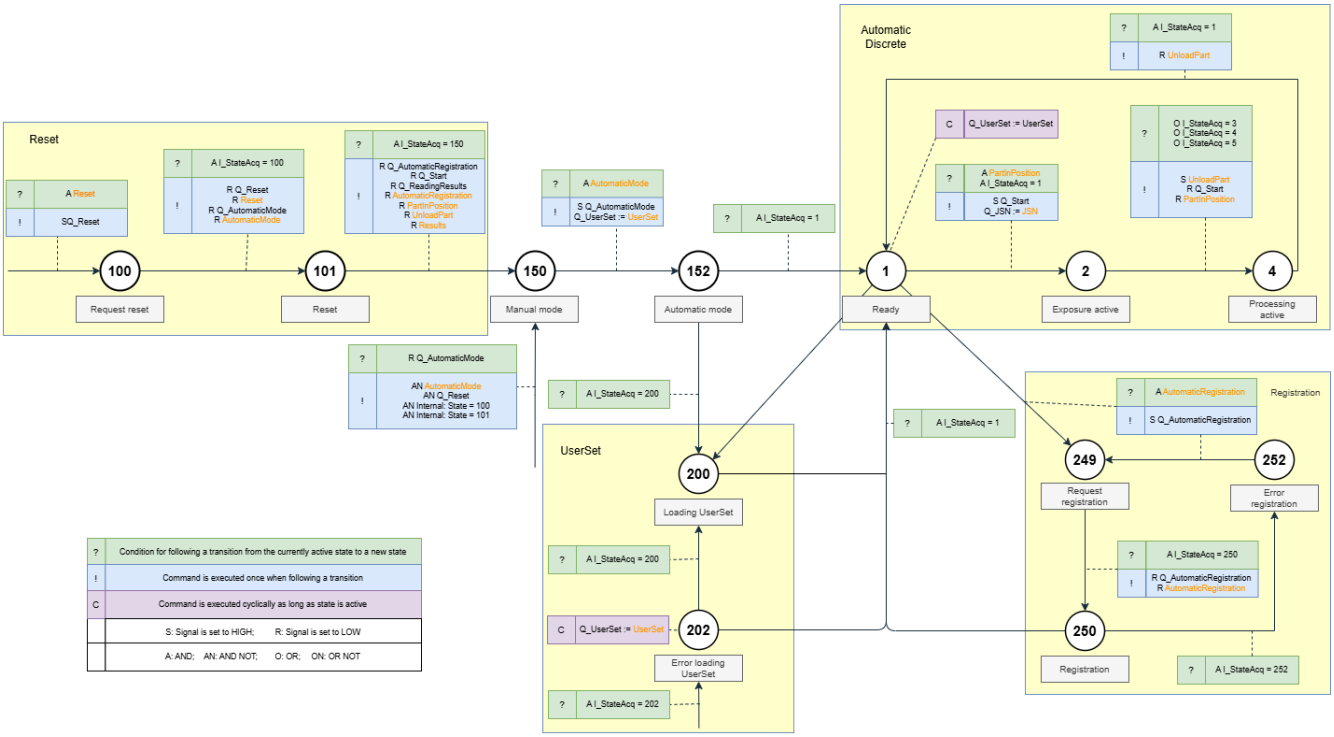


Abbildung 9: Beispielhafte State Machine

Das **Live-Bit** wird mit einer Frequenz von **2 Hz getoggelt** und dient zur Überwachung der Kommunikation. Bleibt das Live-Bit stehen, muss von einer gestörten Kommunikation ausgegangen werden.

Nachfolgend beschriebene Anwendungsfälle

In den folgenden Abschnitten werden typische Signalabläufe beschrieben:

- Reset

- Umschalten zwischen Manual Mode und Automatic Mode
- Laden eines UserSets / Rezepts
- Auslösen einer Registrierung
- Messung im diskreten Betrieb
- Auswertung im diskreten Betrieb
- Messung im kontinuierlichen Betrieb
- Sensorseitige State Machines des MEAutomationCore

Reset-Ablauf aus Sicht des Kommunikationspartners

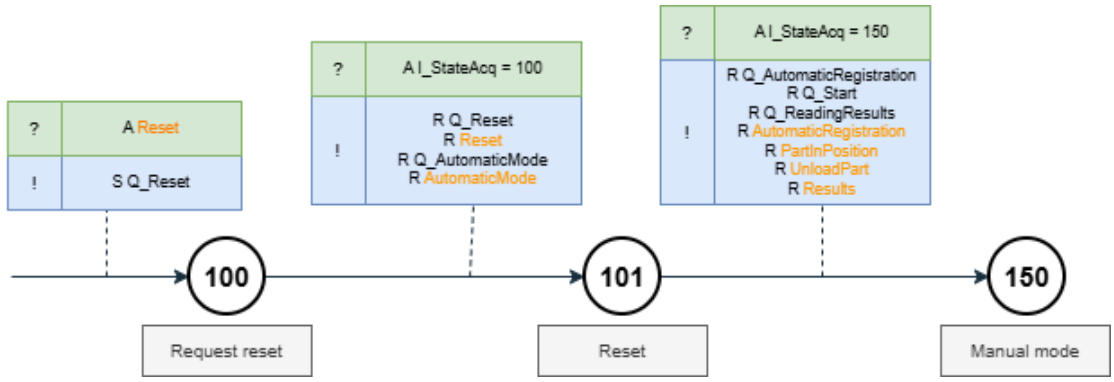


Abbildung 10: Reset-Ablauf aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn die *Reset*-Anforderung durch die Automatisierungsumgebung ausgelöst wird, wird der Ausgang *Q_Reset* gesetzt und in den Status 100 gewechselt.

Der Kommunikationspartner wiederum liest diesen Status zurück und wurde somit bestätigt, dass der Reset-Befehl angekommen ist. Daraufhin kann er sein *Q_Reset* und das *Reset* zurücksetzen und wechselt in Status 101. Zusätzlich wird *Q_AutomaticMode* zurückgesetzt um sicherzustellen, dass der MEAutomationCore im ManualMode verweilt und den Status nicht einfach sofort durchläuft.

Wenn der Kommunikationspartner den Status *I_StateAcq = 150* liest, weiß er, dass der MEAutomationCore den Reset komplett durchlaufen hat und setzt auch seine relevanten Signale zurück, um dann in den Manual Mode zu wechseln.

MEAutomationCore

Dieses Signal wird vom MEAutomationCore gelesen und wenn er nicht bereits im Status 100 ist, wechselt er in diesen.

Der MEAutomationCore wiederum sieht, dass das Reset-Signal zurückgesetzt wurde und wechselt daraufhin in Status 101, zudem setzt er seine relevanten Signale des Interfaces zurück. Sobald die internen Initialisierungsprozesse abgeschlossen wurden, wechselt der MEAutomationCore in den Status 150.

Dieses Verhalten gilt sowohl für den kontinuierlichen als auch für den diskreten Fall. Im diskreten Fall ist zusätzlich die Evaluation State Machine zu betrachten.

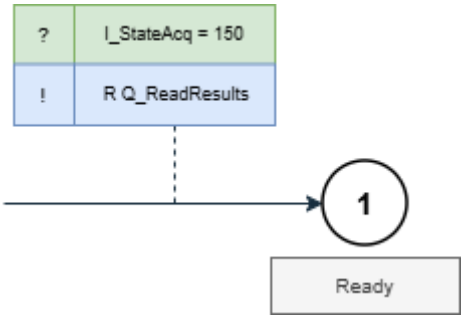


Abbildung 11: Reset Signalablauf der Evaluation State Machine des Kommunikationspartners

Da bei einem Reset auch immer der Manual Mode durchlaufen wird, prüft der Kommunikationspartner auf den *I_StateAcq = 150. Liegt dieser vor,* wird das *Q_ReadResults* Signal zurückgesetzt und zurück in den Evaluation State 1 gesprungen .

Betriebsart wechseln: Manual Mode / Automatic Mode

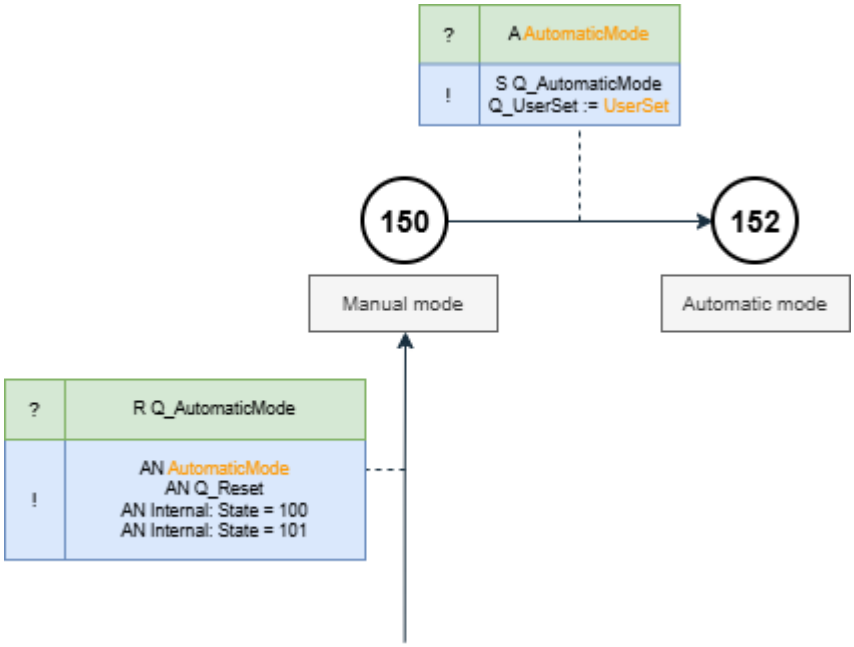


Abbildung 12: Umschalten der Betriebsart in der Acquisition-State Machine aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn der *AutomaticMode* zurückgesetzt werden soll und sich die State Machine nicht gerade in einer Reset-Routine befindet oder diese angefordert wird, wird der Ausgang *Q_AutomaticMode* zurückgesetzt und in den Status 150 „Manual Mode“ gewechselt.

MEAutomationCore

Wenn das Steuerbit für den AutomaticMode zurückgesetzt wurde und sich die State Machine nicht gerade in einer Reset-Routine befindet oder diese angefordert wird in den Status 150 „Manual Mode“ gewechselt. Dabei wird auch das *Q_ActualUserSet := 0* gesetzt.
Somit wird sichergestellt, dass nach einem Umschalten in den AutomaticMode das UserSet nochmal neu geladen wird, auch wenn dieses eventuell zuvor bereits geladen war.

Die State Machine verbleibt im Manual Mode bis durch die Automatisierungsumgebung wieder Anforderung für den *AutomaticMode* ausgelöst wird. Daraufhin wird der Ausgang *Q_AutomaticMode* gesetzt und das angeforderte *UserSet* an den Ausgang *Q_UserSet* übergeben.

Durch den Eingang *I_AutomaticMode* wechselt die State Machine in den Status 151 und dann, sobald der interne Prozess zum Umschalten des UserSets abgeschlossen ist, in den Status 152 „Automatic Mode“.

Dieses Verhalten gilt sowohl für den kontinuierlichen als auch den diskreten Fall. Im diskreten Fall ist zusätzlich die Evaluation State Machine zu betrachten (siehe Abbildung 10).

UserSet / Rezept laden

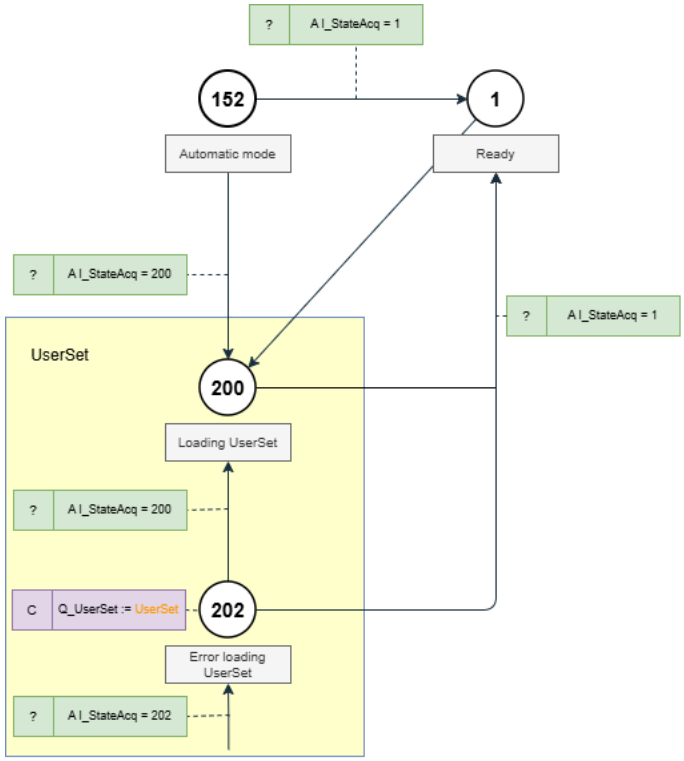


Abbildung 13: UserSet laden aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn der MEAutomationCore nach dem AutomaticMode entweder in Status 1 oder Status 200 gewechselt ist, liest der Kommunikationspartner diesen *I_StateAcq* zurück und wechselt ebenfalls in den entsprechenden Status.

MEAutomationCore

Ausgehend vom Automatic Mode wechselt der MEAutomationCore, wenn kein UserSet übergeben wurde, umgehend in den „Ready“ State 1 oder in Status 200 „Load UserSet“, wenn ein UserSet beim Wechsel in den Automatic Mode übergeben wurde.

Im Status 200 wird zu Beginn das *I_UserSet* in den *Q_ActualUserSet* geschrieben und das Laden durchlaufen. Bei erfolgreichem Abschluss wird in den „Ready“ State 1 gewechselt. Bei einem Fehler wird in den Status 202 gewechselt.

Im Status 200 wird gewartet bis durch den *I_StateAcq* der Abschluss des Ladens erkannt wird und es wird in den entsprechenden Status gewechselt.

Im Status 202 wird das *UserSet* zyklisch an *Q_UserSet* übergeben und dadurch im MEAutomationCore, bei Übergabe eines neuen UserSets, das Laden erneut ausgelöst.
Da im MEAutomationCore theoretisch der Übergang zwischen Status 200 und 202 so schnell erfolgen kann, dass es der Kommunikationspartner nicht mitbekommt, ist der Übergang in den Status 202 beim Kommunikationspartner unabhängig vom vorherigen Zustand.

Bei Betriebsbereitschaft im Ready „State" 1 wird das *UserSet* zyklisch an *Q_UserSet* übergeben und damit das Laden eines neuen UserSets ermöglicht.

Im Fehlerstatus 202 wird solange gewartet, bis ein UserSet übergeben wird, das ungleich des vorherigen UserSets und ungleich 0 ist. Dann wird in Status 200 gewechselt und das Laden erneut probiert.

Aus dem „Ready" State 1 kann auch in Status 200 gewechselt werden, wenn ein neues UserSet ungleich 0 übergeben wird.

Dieses Verhalten gilt sowohl für den kontinuierlichen als auch den diskreten Fall.

Registrierung auslösen

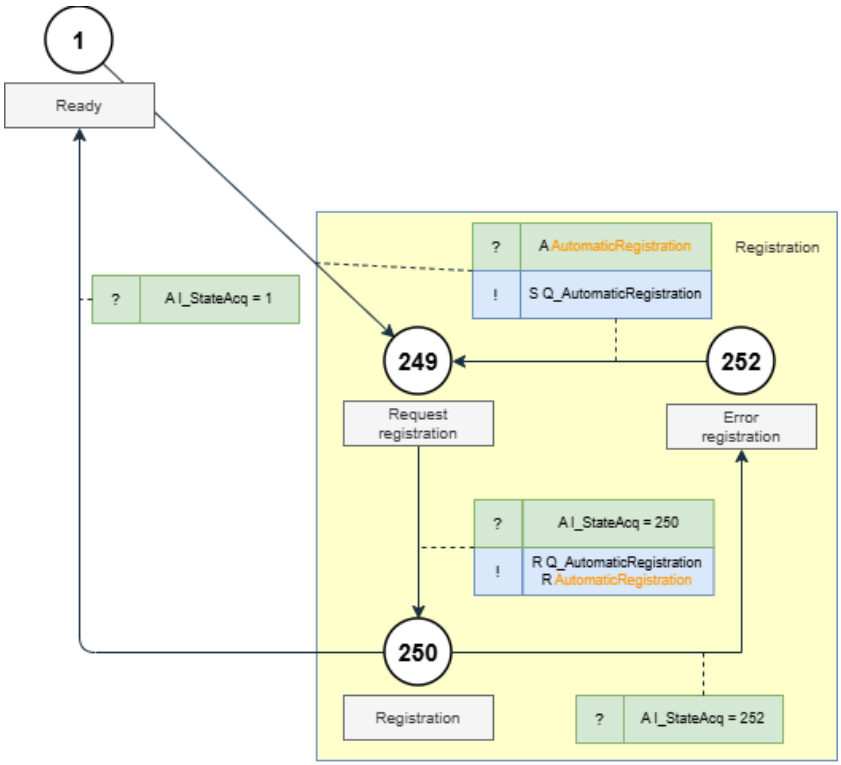


Abbildung 14: Auslösen einer Registrierung aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn die *AutomaticRegistration* Anforderung durch die Automatisierungsumgebung ausgelöst wird, wird der Ausgang *Q_AutomaticRegistration* gesetzt und in den Status 249 gewechselt.

MEAutomationCore

Dieses Signal wird vom MEAutomationCore gelesen woraufhin der MEAutomationCore in den Status 250 wechselt und die Registrierung startet.

Der Kommunikationspartner wiederum liest diesen Status zurück und wurde somit bestätigt, dass der Registrierungsbefehl angekommen ist. Daraufhin kann er sein *Q_AutomaticRegistration* und das *AutomaticRegistration* zurücksetzen und wechselt in Status 250.

Nach Abschluss der Registrierungsroutine wechselt der MEAutomationCore bei erfolgreichem Abschluss wieder zurück in Status 1. Bei einem Fehler während der Routine wird in den Status 252 gewechselt. Voraussetzung für beide Übergänge ist, dass der Eingang *I_PerformRegistration* bereits zurückgesetzt wurde.

Je nach Status des MEAutomationCore (*I_StateAcq)* wechselt der Kommunikationspartner auch entweder in Status 1 oder Status 252. Im Fehlerfall muss die Registrierung erneut angestoßen werden durch die Automatisierungsumgebung (*AutomaticRegistration)* und das Setzen des Ausgangs *Q_AutomaticRegistration*.

Dieses Verhalten gilt sowohl für den kontinuierlichen als auch den diskreten Fall.

Messablauf im diskreten Betrieb

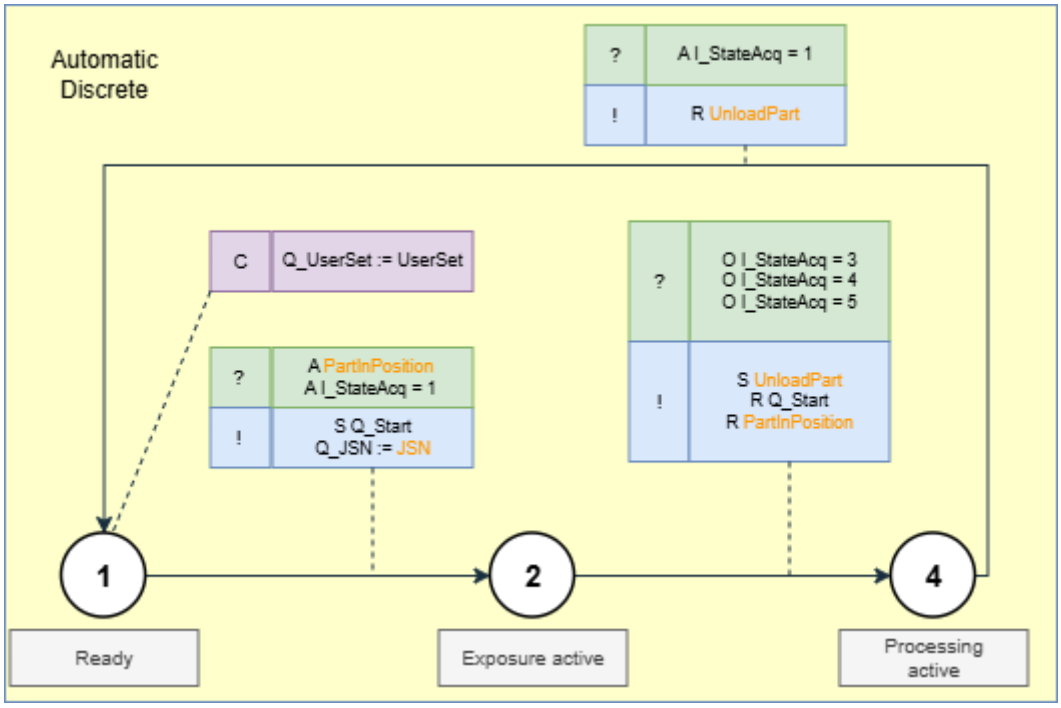


Abbildung 15: Messung im diskreten Fall aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn im Ready Status 1 die Anforderung der Automatisierungsumgebung zu einer Messung mit *PartInPosition* gesetzt wird und sich auch der MEAutomationCore im Ready Status befindet (*I_StateAcq = 1*), dann setzt der Kommunikationspartner den *Q_Start*, schreibt die *JSN* auf den jeweiligen Ausgang *Q_JSN* und wechselt in den Status 2.

MEAutomationCore

Wenn das Startsignal des Kommunikationspartners gelesen wird und das aktuell ausgegebene UserSet dem angeforderten UserSet

entspricht, wird die Messung gestartet und in den Status 2 gewechselt.

Der Status 2 ist aktiv bis die Datenaufnahme abgeschlossen wurde. Anschließend werden die Daten zu einer 3D-Punktewolke im Status 3 verarbeitet und dann an die Evaluation State Machine übergeben (Status 4). Im Status 5 wartet der MEAutomationCore dann wieder auf die Rückmeldung durch den Kommunikationspartner.

Sobald der Kommunikationspartner *I_StateAcq = 3* zurückliest, weiß er, dass die Datenaufnahme abgeschlossen ist. Damit kann er die Bewegungsfreigabe *UnloadPart* setzen und *PartInPosition* und *Q_Start* zurücksetzen. Da auf Seiten des MEAutomationCore der Statusübergang von Status 3 auf Status 5 ohne Interaktion des Kommunikationspartners erfolgt und somit die Status theoretisch übersehen werden können, wird zusätzlich auf *I_StateAcq = 4* oder *I_StateAcq = 5* geprüft.

Mit dem zurückgesetzten Startsignal kann der MEAutomationCore wieder in den „Ready“ Status 1 wechseln.

Der *I_StateAcq = 1* ist die Übergangsbedingung für den Kommunikationspartner für seinen Wechsel in den Status 1. Dabei wird das Signal an die Automatisierungsumgebung *UnloadPart* wieder zurückgesetzt.

Im diskreten Fall muss zusätzlich muss die Evaluation State Machine betrachtet werden.

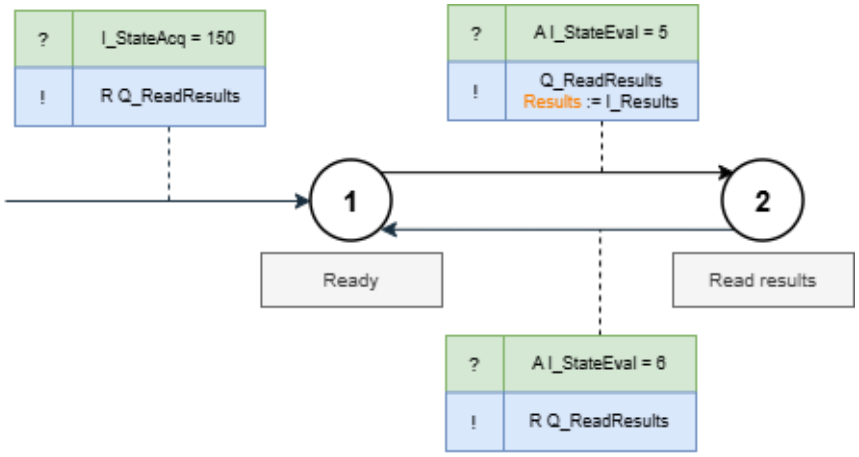


Abbildung 16: Signalverlauf der Evaluation State Machine aus Sicht des Kommunikationspartners

Kommunikationspartner

Der Statusübergang des MEAutomationCore (*I_StateEval = 5)* ist die Übergangsbedingung für den Kommunikationspartner. Dabei übergibt sie die Messergebnisse aus *Q_ReadResults* an die Automatisierungsumgebung (*Results*) und setzt dabei *Q_ReadResults* als Signal, dass die Messergebnisse abgeholt wurden.

MEAutomationCore

Nach dem Acquisition State 4 werden die Daten an die Evaluation State Machine übergeben um dort ausgewertet zu werden.
Mit der Übergabe der Daten wechselt die Evaluation State Machine von Status 1 in den Status 4 und setzt dabei *Q_Results* zurück. Sobald die Auswertung abgeschlossen ist wechselt die Evaluation State Machine in Status 5 und schreibt seine Ergebnisse in *Q_Results*.

Mit dem *I_StateEval = 6* wechselt der Kommunikationspartner zurück in Status 1 und setzt dabei sein *Q_ReadResults* wieder zurück.

Das Signal *I_ReadResults* vermittelt dem MEAutomationCore, dass die Messergebnisse abgeholt werden und er wechselt in Status 6.

Der MEAutomationCore weiß, dass das zurückgesetzte *I_ReadResults*, dass der Vorgang abgeschlossen ist und kann wieder in seinen Status 1 zurückwechseln.

Messablauf im kontinuierlichen Betrieb

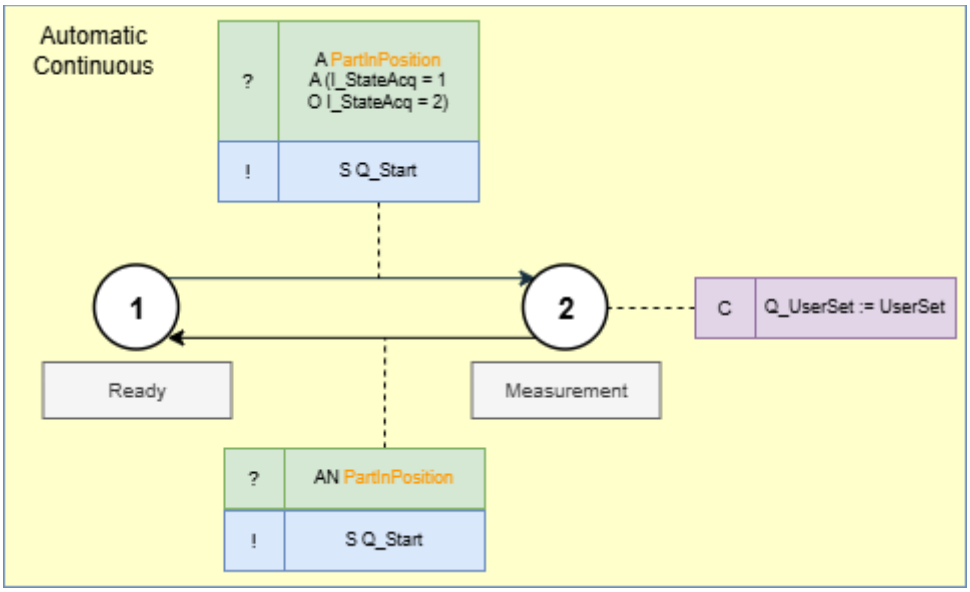


Abbildung 17: Messung im Kontinuierlichen Fall aus Sicht des Kommunikationspartners

Kommunikationspartner

Wenn im Ready Status 1 die Anforderung der Automatisierungsumgebung zu einer Messung mit *PartInPosition* gesetzt wird und sich auch der MEAutomationCore im Ready Status befindet (*I_StateAcq = 1*), dann setzt der Kommunikationspartner den *Q_Start* und wechselt in Status 2.

Die Messergebnisse (*I_Results*) können kontinuierlich gelesen und an die Automatisierungsumgebung (*Results*) übergeben werden. Wenn *PartInPosition* zurückgesetzt wird, setzt der Kommunikationspartner das *Q_Start* zurück und wechselt in Status 1.

MEAutomationCore

Wenn das Startsignal des Kommunikationspartners gelesen wird und das aktuell ausgegebene UserSet dem angeforderten UserSet entspricht wird die kontinuierliche Messung und Verarbeitung gestartet und in den Status 2 gewechselt.

Sobald das *I_Start* Signal entfällt, wird die Messung abgebrochen und in den Status 1 zurück gewechselt.

3. Anhang C Server State Machines (sensor-seitig)

Um es im diskreten Fall zu ermöglichen, mit der nächsten Messung zu starten während die Vorhergehende noch ausgewertet wird, gibt es zwei State Machines (Acquisition und Evaluation).

Diskreter Fall:

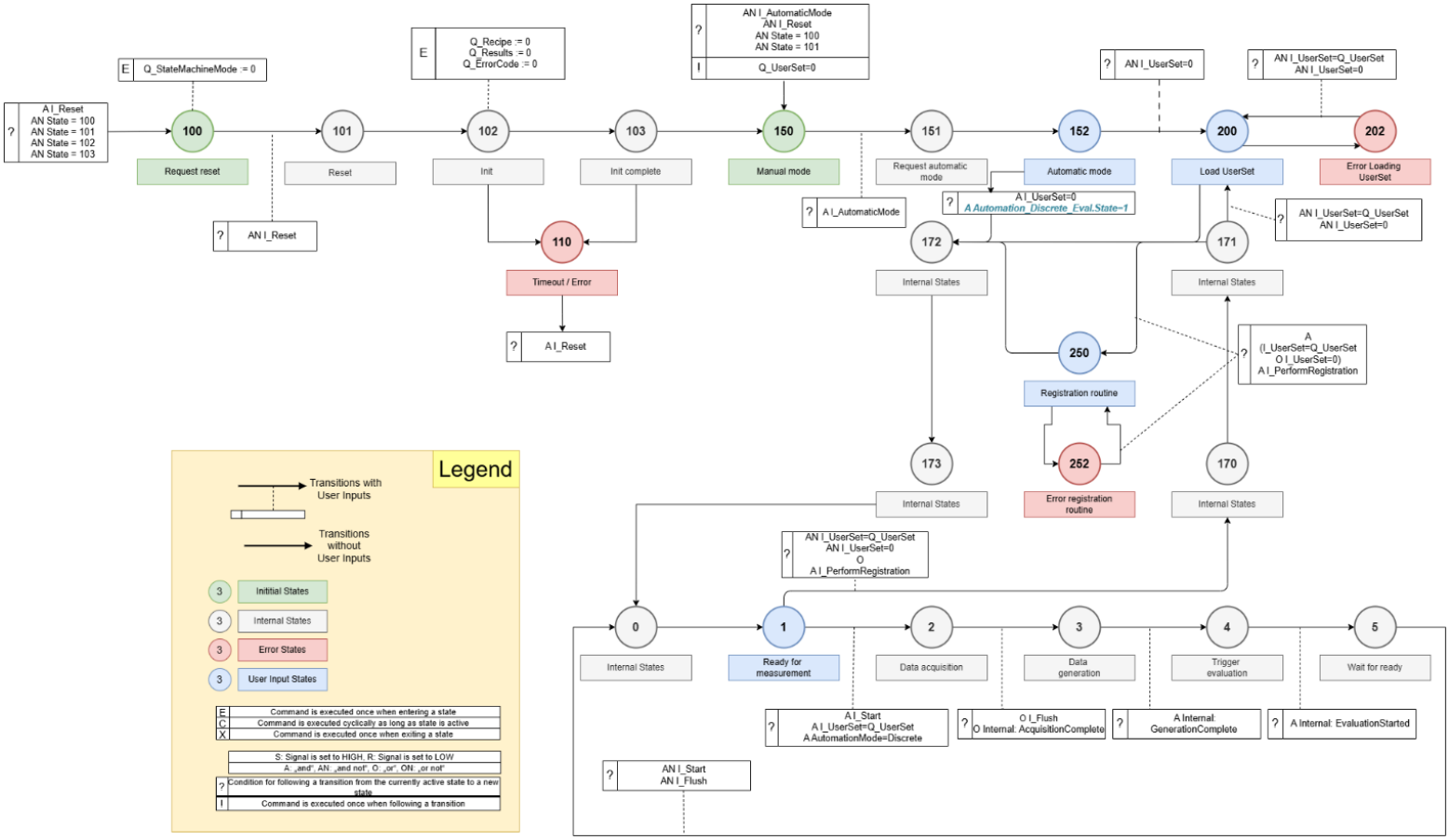


Abbildung 18: MEAutomationCore State Machine for discrete acquisition

Obiges Diagramm zeigt die State Machine „**Acquisition**" (Datenaufnahme), die auf dem Micro-Epsilon Sensor im **diskreten Fall** abläuft.

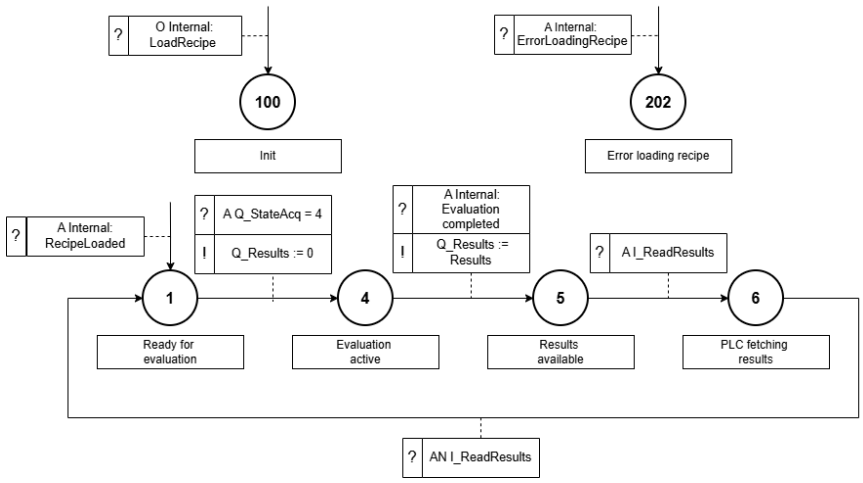


Abbildung 19: MEAutomationCore discrete evaluation

Obiges Diagramm zeigt die State Machine „**Evaluation**“ (Datenauswertung), die auf dem Micro-Epsilon Sensor im **diskreten Fall** abläuft.

Kontinuierlicher Fall

Auch im kontinuierlichen Fall laufen Acquisition und Evaluation State Machine parallel.

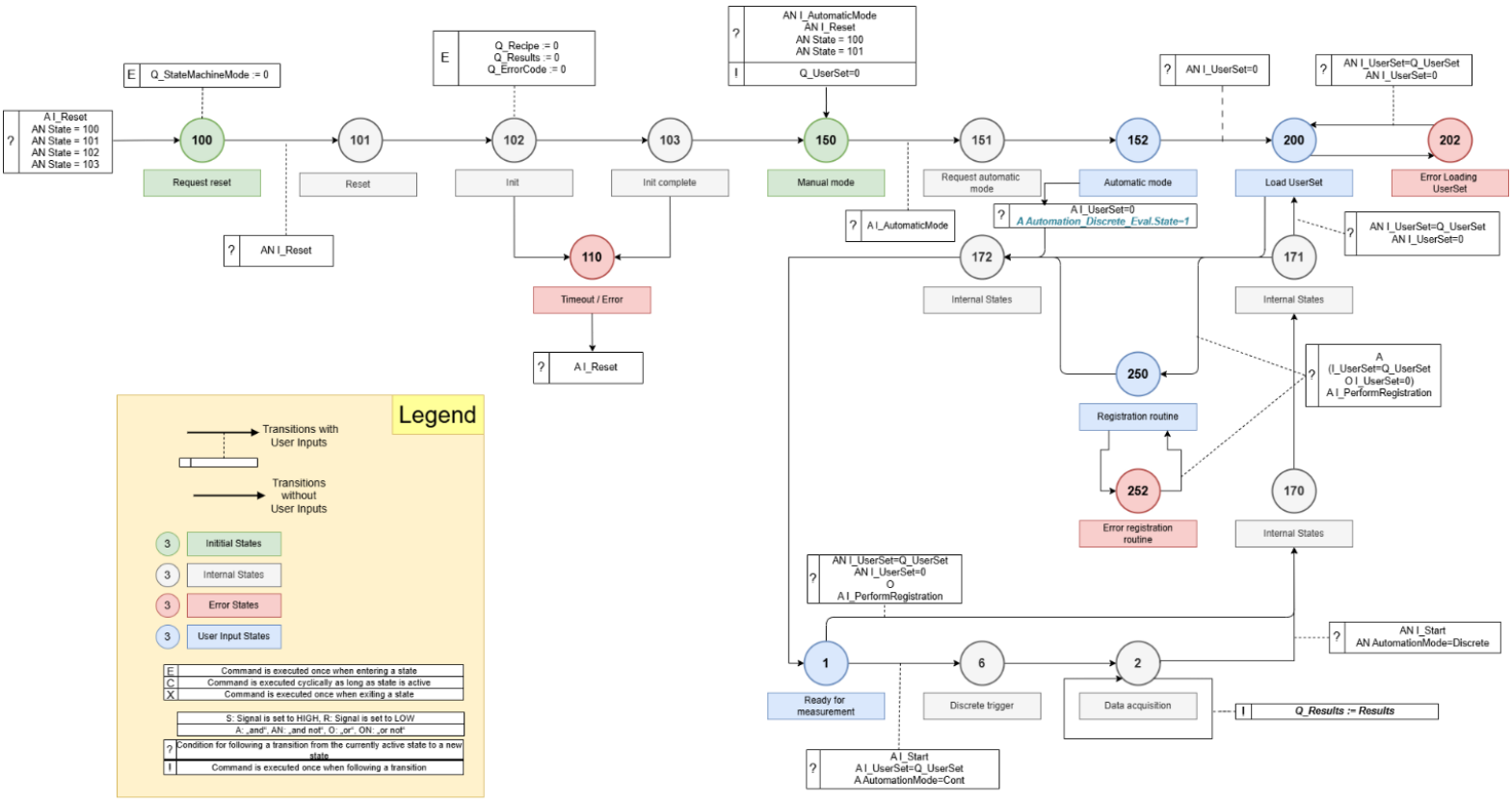


Abbildung 20: MEAutomationCore continuous acquisition

Obiges Diagramm zeigt die State Machine „**Acquisition**“ (Datenaufnahme), die auf dem Micro-Epsilon Sensor im **kontinuierlichen Fall** abläuft.

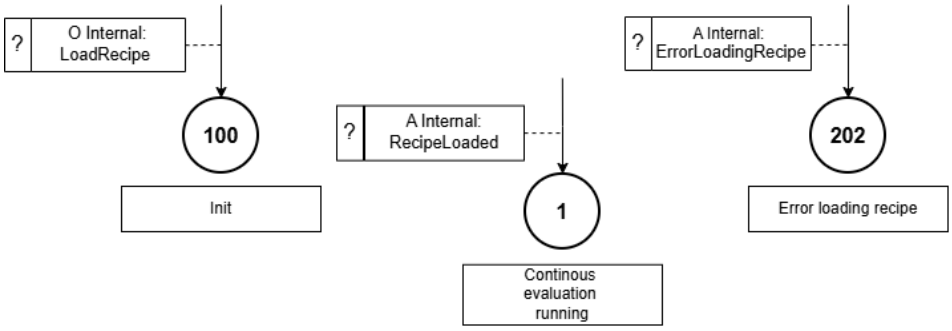


Abbildung 21: MEAutomationCore continuous evaluation

Obiges Diagramm zeigt die State Machine „**Evaluation**“ (Datenauswertung), die auf dem Micro-Epsilon Sensor im **kontinuierlichen Fall** abläuft. In diesem Sonderfall finden sich ausschließlich unbedingte Zustandsübergänge.

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/

X9750515-A022106SDR

© MICRO-EPSILON MESSTECHNIK