SIMOTION – Stationsausfälle und Modulzustände diagnostizieren

SIMOTION

Applikationsbeschreibung • Juni 2013

Applikationen & Tools

Answers for industry.

SIEMENS

Siemens Industry Online Support

Dieser Beitrag stammt aus dem Siemens Industry Online Support. Durch den folgenden Link gelangen Sie direkt zur Downloadseite dieses Dokuments:

http://support.automation.siemens.com/WW/view/de/53705461

Vorsicht:
Die in diesem Beitrag beschriebenen Funktionen und Lösungen beschränken sich überwiegend auf die Realisierung der Automatisierungsaufgabe. Bitte beachten Sie darüber hinaus, dass bei Vernetzung Ihrer Anlage mit anderen Anlagenteilen, dem Unternehmensnetz oder dem Internet entsprechende Schutzmaßnahmen im Rahmen von Industrial Security zu ergreifen sind. Weitere Informationen dazu finden Sie unter der Beitrags-ID 50203404.

http://support.automation.siemens.com/WW/view/de/50203404

S

SIMOTION
Stationsausfälle und Modulzustände diagnostizieren

Gewährleistung und Haftung

Hinweis

Die Applikationsbeispiele sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Applikationsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern sollen lediglich Hilfestellung bieten bei typischen Aufgabenstellungen. Sie sind für den sachgemäßen Betrieb der beschriebenen Produkte selbst verantwortlich. Diese Applikationsbeispiele entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieser Applikationsbeispiele erkennen Sie an, dass wir über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden können. Wir behalten uns das Recht vor, Änderungen an diesen Applikationsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesem Applikationsbeispiel und anderen Siemens Publikationen, wie z.B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Für die in diesem Dokument enthaltenen Informationen übernehmen wir keine Gewähr.

Unsere Haftung, gleich aus welchem Rechtsgrund, für durch die Verwendung der in diesem Applikationsbeispiel beschriebenen Beispiele, Hinweise, Programme, Projektierungs- und Leistungsdaten usw. verursachte Schäden ist ausgeschlossen, soweit nicht z.B. nach dem Produkthaftungsgesetz in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der Verletzung des Lebens, des Körpers oder der Gesundheit, wegen einer Übernahme der Garantie für die Beschaffenheit einer Sache, wegen des arglistigen Verschweigens eines Mangels oder wegen Verletzung wesentlicher Vertragspflichten zwingend gehaftet wird. Der Schadensersatz wegen Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegt oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit zwingend gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist hiermit nicht verbunden.

Inhaltsverzeichnis

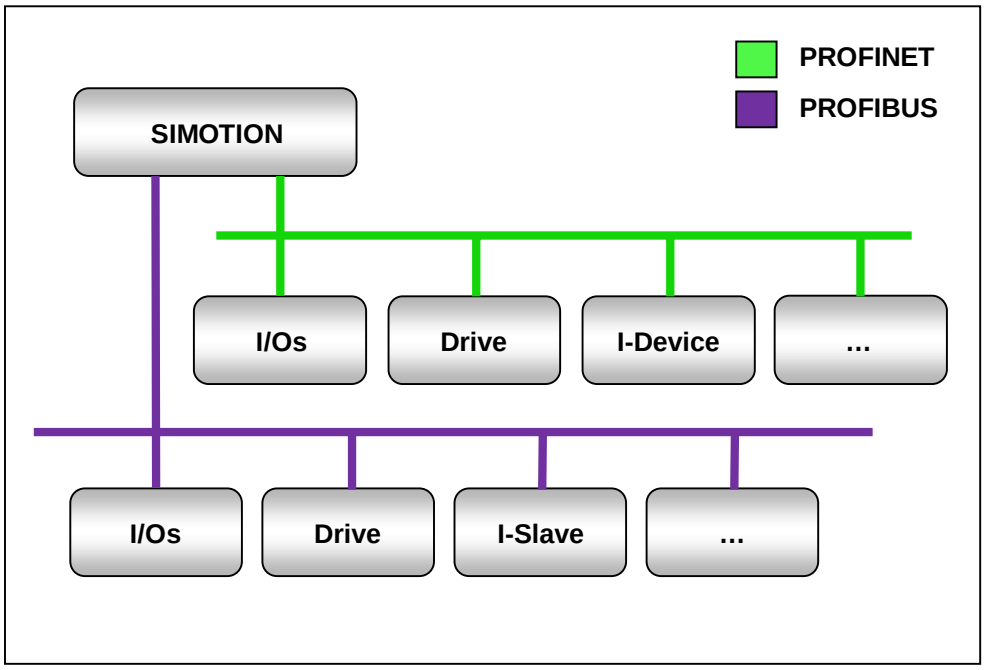
1 Aufgabe

1.1 Übersicht

Überblick über die Automatisierungsaufgabe

Folgendes Bild gibt einen Überblick über die Automatisierungsaufgabe.

Abbildung 1-1



Beschreibung der Automatisierungsaufgabe

Die an einer SIMOTION Steuerung angeschlossenen IO-Devices sollen auf Ausfälle und Modulzustandsänderungen überwacht werden. Dies soll zur Laufzeit der SIMOTION Applikation realisiert werden.

Es sollen dabei sowohl die Zustandsdaten für PROFINET IO-Devices als auch PROFIBUS DP-Slaves in einer Struktur für den Anwender hinterlegt werden.

In der Struktur, die nach an der SIMOTION Steuerung konfigurierten Bussegmenten aufgeteilt ist, sollen alle Stationen sowie deren Module aufgelistet werden. Der Anwender soll dort für die jeweilige projektierte Station sehen, ob ein Fehler an einem oder mehreren Modulen vorliegt, oder ob die Station ausgefallen ist.

Die Fehlerinformationen sollen ebenfalls in der Struktur stationszugehörig hinterlegt und eine bestimmte Anzahl an Fehlern gespeichert werden, um den Fehlerverlauf nachvollziehen zu können.

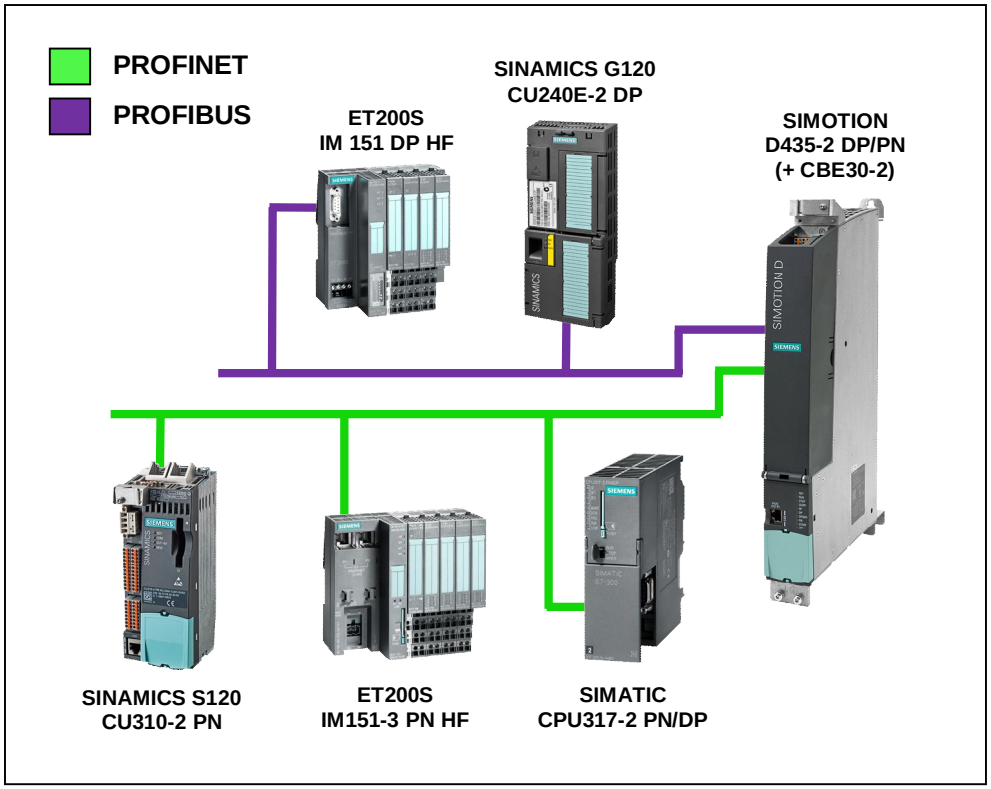
2 Lösung

2.1 Übersicht Gesamtlösung

Schema

Die folgende Abbildung zeigt schematisch die wichtigsten Komponenten der Lösung:

Abbildung 2-1



Vorteile

Die vorliegende Applikation bietet Ihnen folgende Vorteile:

- Performante Diagnose von PROFINET IO-Devices sowie PROFIBUS DP-Slaves einer SIMOTION Steuerung bis auf Modulebene
- Einmalige Ermittlung der aktuellen Zustände der Stationen zur Laufzeit der SIMOTION Applikation
- Speicherung der Diagnosedaten in einer Anwenderstruktur
- Automatische Aktualisierung der Anwenderstruktur mittels Fehlerinformationen aus der `PeripheralFaultTask`
- Zugriff auf die Diagnosedaten von HMIs aus möglich

2.2 Beschreibung der Kernfunktionalität

Dieses Applikationsbeispiel zeigt eine Möglichkeit der einfachen Diagnose von Stationsausfällen oder Zustandsänderungen einzelner Module einer Station.

Es werden die von einer SIMOTION Steuerung betriebenen und in der Hardwarekonfiguration projektierten PROFINET IO-Devices und PROFIBUS DP-Slaves diagnostiziert (inkl. I-Devices / I-Slaves).

Zur Laufzeit der Applikation werden einmalig die Ist-Zustände aller projektierten Stationen (sowie deren Module) ermittelt und diese in einer Anwenderstruktur hinterlegt. Die Aktualisierung der Struktur erfolgt danach durch Auswertung der sogenannten „Task-Start-Informationen (TSI)“ der `PeripheralFaultTask` und Kopieren der Fehlerinformationen an die entsprechende Stelle in der Anwenderstruktur.

Die ST-Quellen, die für die Diagnose benötigt werden, sind im mitgelieferten ZIP-Archiv `53705461_SIMOTION_Diagnostics_V2_0.zip` enthalten.

Verwendete Systemfunktionen

Die folgenden von SIMOTION bereitgestellten Systemfunktionen werden im Applikationsbeispiel verwendet:

- **_getSegmentIdentification()**
  Alle projektierten Bussegmente werden ermittelt.
- **_getStateOfAllDpStations()**
  Der Status aller projektierten Devices wird ermittelt.
- **_getNextLogAddress()**
  Alle projektierten logischen Adressen eines Bussegments werden ermittelt.

Verwendete Konstanten

Im Applikationsbeispiel werden globale Konstanten verwendet, die vom Anwender angepasst werden können, um die Diagnose weiter zu optimieren.

Die folgenden Konstanten befinden sich in der ST-Quelle `fMachineNetworkDiagnostics`:

- **NUMBER_OF_NETWORKS**
  Anzahl an Bussegmenten, die diagnostiziert werden sollen
  (Default: 5, Maximal: 8).
- **HIGHEST_STATION_ADDRESS_AT_NETWORK**
  Größte Stationsadresse aller projektierten Bussegmente. Alle Stationen mit einer Stationsadresse kleiner gleich dieser Konstanten werden diagnostiziert.
  (Default: 128, Maximal: 128)
- **HIGHEST_NUMBER_OF_MODULES_PER_STATION**
  Größte Anzahl an Modulen (≙ E/A-Adressen) pro Station aller projektierten Bussegmente, deren Status ermittelt werden soll. Für alle Stationen mit einer Modulanzahl (≙ E/A-Adressen) kleiner gleich dieser Konstanten werden die Modulzustände diagnostiziert.
  (Default: 64, Maximal: 255)
- **NUMBER_OF_STORED_EVENTS_PER_STATION**
  Anzahl an Ereignissen, die pro Station gespeichert werden sollen.
  (Default: 8, Maximal 255)

Übersicht der Quellen / Bausteine

Tabelle 2-1

Quellen	Programmiersprache	Know How Schutz
pMachineNetworkDiagnostics	ST	Nein
fMachineNetworkDiagnostics	ST	Nein

Funktion / Funktionsbaustein	Eigenschaft	Anpassung an Applikation erforderlich
FBGetStateOfAllStationsAtNetwork	Dieser Funktionsbaustein ermittelt die aktuellen Zustände aller projektierten Stationen.	Nein
FBInitNetworkDiagnosticsStructure	Dieser Funktionsbaustein dient zur Initialisierung der Diagnose-Struktur.	Nein
FCMachinePeripheralFailureHandling	Diese Funktion dient zur Auswertung der Task-Start-Informationen der `PeripheralFaultTask`.	Nein

2.3 Verwendete Hard- und Software-Komponenten

Das Applikationsbeispiel wurde mit den nachfolgenden Komponenten erstellt:

Hardware-Komponenten

Tabelle 2-2

Komponente	Anz.	Bestellnummer	Hinweis
SIMOTION D435-2 DP/PN	1	6AU1 435-2AD00-0AA0	V 4.3 SP1 HF12
SIMATIC 317-2 PN/DP (I-Device)	1	6ES7 317-2EK14-0AB0	V3.2.7
ET200S HF (PROFINET)	1	6ES7 151-3BA23-0AB0	V 7.0.1
SINAMICS S120 CU310-2 PN	1	6SL3 040-1LA01-0AA0	V 4.6 HF3
ET200S HF (PROFIBUS)	1	6ES7 151-1BA00-0AB0	V1.1.3
SINAMICS G120 CU240E-2 DP	1	6SL3 244-0BB12-1PA1	V4.6 HF3

Standard Software-Komponenten

Tabelle 2-3

Komponente	Anz.	Bestellnummer	Hinweis
STEP7	1	6ES7 810-4CC10-0YA7	V5.5 SP3
SIMOTION SCOUT	1	6AU1 810-1BA43-1XA0	V4.3 SP1 HF12
SSP SINAMICS	1	---	V4.6 (Download)
SSP SINAMICS G120	1	---	V4.6 (Download)

Beispieldateien und Projekte

Die folgende Liste enthält alle Dateien und Projekte, die in diesem Beispiel verwendet werden.

Tabelle 2-4

Komponente	Hinweis
53705461_SIMOTION_Diagnostics_V2_0_de.pdf	Dieses Dokument
53705461_SIMOTION_Diagnostics_V2_0.zip	ZIP-Archiv mit den für die Diagnose benötigten ST-Quellen

3 Funktionsmechanismen dieser Applikation

3.1 Anwenderstruktur „gasMachineNetwork“

Die Anwenderstruktur `gasMachineNetwork` besteht aus einem Array der Struktur `sNetworkType` (siehe Tabelle 3-1) mit einer dynamischen Anzahl an Elementen für die einzelnen Netzwerke einer SIMOTION Steuerung. Die Größe des Arrays ist dabei abhängig vom Wert der Konstanten `NUMBER_OF_NETWORKS`.

In dieser Struktur werden alle für den Anwender relevanten Daten abgelegt, die zur Diagnose der Netzwerke einer SIMOTION Steuerung sowie deren Stationen benötigt werden.

Struktur `sNetworkType`

Tabelle 3-1[1]

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sNetwork Type						
	eTypeOf Network	Enum Network Type	OUT	---	UNKNOWN	Anzeige des Netzwerktyps
	u8NumberOf StationsAt Network	USINT	OUT	---	0	Anzahl der projektierten Stationen im Netzwerk
	u8NumberOf Disturbed Stations	USINT	OUT	---	0	Anzahl der aktuell fehlerhaften Stationen im Netzwerk
	u8Last Disturbed Station	USINT	OUT	---	0	Anzeige der letzten fehlerhaften Station im Netzwerk
	asStation	ARRAY [1.`HIGHEST_`...] OF sStateOf DevicesAt NetworkType	OUT	---	---	Array der Struktur mit dynamischer Anzahl an Elementen für die Stationen im Netzwerk; Arraygröße abhängig vom Wert der Konstanten. (siehe Tabelle 3-2)

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

[1] ARRAY [1.`HIGHEST_STATION_ADDRESS_AT_NETWORK`] OF sStateOfDevicesAtNetworkType

Struktur `sStateOfDevicesAtNetworkType`

Tabelle 3-2[2]

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sStateOf DevicesAt Network Type						
	i32 Diagnostic Address	DINT	OUT	---	0	Diagnoseadresse der Station
	eActualState	Enum StateOf DpStation	OUT	---	NO_VALID _STATE	Aktueller Zustand der Station
	u32 InterruptId	UDINT	OUT	---	0	Fehler, der für diese Station in der `Peripheral-FaultTask` gemeldet wurde
	b16Event Class	WORD	OUT	---	16#00_00	Ereignisklasse (abhängig von `u32Interrupt-Id`) als weitere Fehlerinformation
	b16FaultId	WORD	OUT	---	16#00_00	Störungskennung (abhängig von `u32Interrupt-Id` und `b16EventClass`) als weitere Fehlerinformation
	b32Details	DWORD	OUT	---	16#00_00 _00_00	Detailinformationen (abhängig von `u32Interrupt-Id` und `b16EventClass`) zum vorliegenden Fehler
	u8NumberOf Disturbed Modules	USINT	OUT	---	0	Anzahl der aktuell fehlerhaften Module (≙ E/A-Adressen) der Station
	u8Last Disturbed Module	USINT	OUT	---	0	Nummer des letzten fehlerhaften Moduls der Station
	sEvent History	sEvent History Type	OUT	---	---	Struktur, die zur Speicherung der Fehler an der Station dient (siehe Tabelle 3-3)

[2] ARRAY [1..`HIGHEST_NUMBER_OF_MODULES_PER_STATION`] OF sModuleType

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sStateOf DevicesAt Network Type						
	asModule	ARRAY [1..HIGHEST_...] OF sModuleType	OUT	---	---	Array der Struktur mit dynamischer Anzahl an Elementen für die Module (≙ E/A-Adressen) der Station; Arraygröße abhängig vom Wert der Konstanten (siehe Tabelle 3-5)

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter
2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

Struktur `sEventHistoryType`

Tabelle 3-3[3]

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sEvent HistoryType						
	u16Last EventIndex	UINT	OUT	---	0	Zeiger auf den letzten Fehlereintrag im Array `asEvent`, das als Ringpuffer dient
	asEvent	ARRAY [1..NUMBER_...] OF sEventType	OUT	---	---	Array der Struktur mit dynamischer Anzahl an Elementen zur Speicherung der Fehler der Station; Größe des Arrays abhängig vom Wert der Konstanten (siehe Tabelle 3-4)

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter
2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

[3] ARRAY [1..NUMBER_OF_STORED_EVENTS_PER_STATION] OF sEventType

Struktur `sEventType`

Tabelle 3-4

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sEvent Type						
	dtEvent Occured	DT	OUT	---	DT#0001-01-01-0:0:0	Zeitpunkt des Aufrufs der `Peripheral-FaultTask` (d.h. Zeitpunkt des kommenden / gehenden Fehlers)
	u32 InterruptId	UDINT	OUT	---	0	Fehler, der für diese Station in der `Peripheral-FaultTask` gemeldet wurde
	b16Event Class	WORD	OUT	---	16#00_00	Ereignisklasse (abhängig von `u32InterruptId`) als weitere Fehlerinformation
	b16FaultId	WORD	OUT	---	16#00_00	Störungskennung (abhängig von `u32InterruptId` und `b16EventClass`) als weitere Fehlerinformation
	b32Details	DWORD	OUT	---	16#00_00_00_00	Detailinformationen (abhängig von `u32InterruptId` und `b16EventClass`) zum vorliegenden Fehler
	u8Disturbed Module	USINT	OUT	---	0	Nummer des Moduls, an dem der Fehler aufgetreten ist

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

Struktur `sModuleType`

Tabelle 3-5

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sModule Type						
	i32Logical Address	DINT	OUT	---	0	Logische Adresse des Moduls
	eAddress Type	EnumModule Type	OUT	---	NO_VALID _TYPE	Typ der logischen Adresse
	u32Length OfAddress	UDINT	OUT	---	0	Länge der Moduladresse in Byte
	u32 InterruptId	UDINT	OUT	---	0	Fehler, der für dieses Modul in der `Peripheral-FaultTask` gemeldet wurde
	b16Event Class	WORD	OUT	---	16#00_00	Ereignisklasse (abhängig von `u32InterruptId`) als weitere Fehlerinformation
	b16FaultId	WORD	OUT	---	16#00_00	Störungskennung (abhängig von `u32InterruptId` und `b16EventClass`) als weitere Fehlerinformation
	b32Details	DWORD	OUT	---	16#00_00 _00_00	Detailinformationen (abhängig von `u32InterruptId` und `b16EventClass`) zum vorliegenden Fehler

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

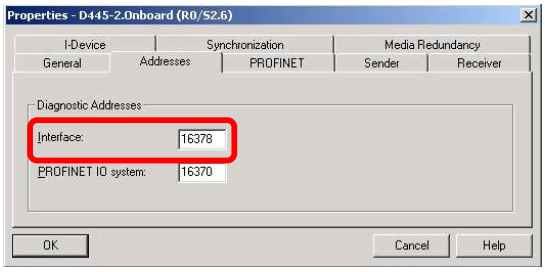
2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

3.2 FBGetStateOfAllStationsAtNetwork

Der Funktionsbaustein `FBGetStateOfAllStationsAtNetwork` liefert die Zustände aller projektierten Stationen eines Netzwerks. Es werden sowohl PROFIBUS DP als auch PROFINET IO Teilnehmer diagnostiziert.

3.2.1 Funktionalität

- Der Baustein ist für den zyklischen Betrieb programmiert.
- Über eine positive Flanke am Eingang `execute` wird die Funktionalität gestartet.
- Der Baustein wird mit der Angabe der Diagnoseadresse (`diagAddressOfNetwork`) des entsprechenden Netzwerks aufgerufen.

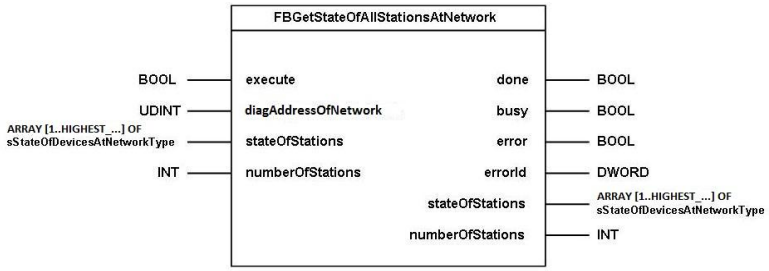


- Die Zustände aller an diesem Netzwerk projektierten Stationen werden ausgelesen.
- Es wird intern die Systemfunktion `_getStateOfAllDpStations()` aufgerufen.
- Tritt beim Auslesen der Stationen ein Fehler auf, wird dies am Ausgang `error` angezeigt. Es kann dann mit der dazu ausgegebenen `errorId` die Fehlerursache nachvollzogen werden.

  Dabei sind die verschiedenen Error-ID's in der Online Hilfe von SIMOTION zur Systemfunktion `_getStateOfAllDpStations()` nachzulesen.
- Der Ausgang `done` zeigt die erfolgreiche Abarbeitung des Funktionbausteins an. Nur bei `done = TRUE` stehen in der Anwenderstruktur `gasMachineNetwork` die Ergebnisse des Funktionsbausteins an.

3.2.2 Schematische KOP Darstellung

Abbildung 3-1[4]



[4] ARRAY [1..HIGHEST_STATION_ADDRESS_AT_NETWORK] OF sStateOfDevicesAtNetworkType

3.2.3 Parameter

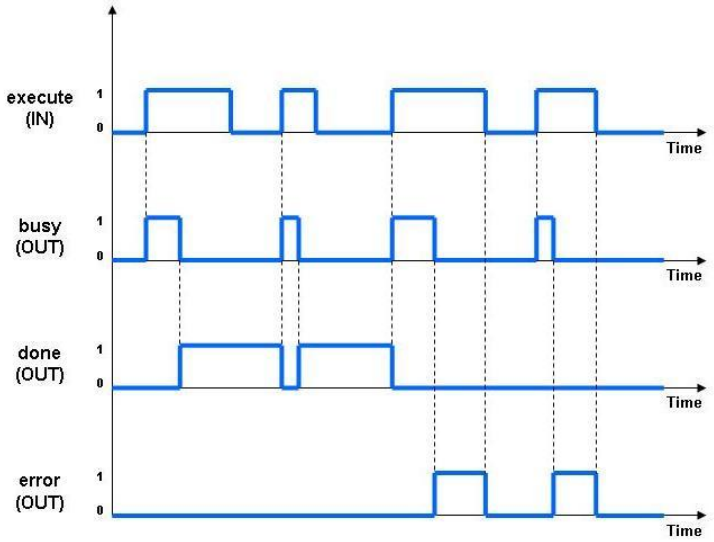
Tabelle 3-6[5]

Element	P-Typ[1)]	Datentyp	M/O[2)]	Initialwert	Bedeutung
execute	IN	BOOL	M	FALSE	TRUE: Start der Funktionalität
diagAddressOf Network	IN	UDINT	M	0	Diagnoseadresse des zu diagnostizierenden Netzwerkes
stateOfStations	IN/OUT	ARRAY [1..HIGHEST_...] OF sStateOfDevices AtNetworkType	M	-	Übergabestruktur für Status der Stationen und deren Module (siehe auch Tabelle 3-2)
numberOf Stations	IN/OUT	INT	M	0	Übergabestruktur für Anzahl der Stationen im Netzwerk
done	OUT	BOOL	-	FALSE	Auftrag in Bearbeitung
busy	OUT	BOOL	-	FALSE	Bearbeitung abgeschlossen
error	OUT	BOOL	-	FALSE	TRUE: Fehler beim Aufruf der Funktion
errorId	OUT	DWORD	-	16#00_00 _00_00	Fehlercode

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter
2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

3.2.4 Zeitablaufdiagramm

Abbildung 3-2



[5] ARRAY [1..HIGHEST_STATION_ADDRESS_AT_NETWORK] OF sStateOfDevicesAtNetworkType

Bausteinabarbeitung ohne Fehler

- Der Baustein wird mit einer steigenden Flanke am Eingang `execute` gestartet.
- Mit dem Start zeigt der Baustein seine aktive Bearbeitung mit `busy = TRUE` an.
- Wenn die Bausteinbearbeitung wieder abgeschlossen ist, wird der Ausgang `busy = FALSE`.
- Gleichzeitig meldet der Baustein über `done = TRUE`, dass die Bausteinfunktionalität erfolgreich abgeschlossen wurde.
- Wird `execute` vor Beendigung der Funktionalität erneut von FALSE auf TRUE gesetzt, wird die aktuelle Bearbeitung abgebrochen und der Baustein beginnt wieder mit der Abarbeitung der Funktionalität.
- Es werden alle Ausgänge initialisiert.

Bausteinabarbeitung mit Fehler

- Der Baustein wird mit einer steigenden Flanke an `execute` gestartet.
- Mit dem Start zeigt der Baustein seine aktive Bearbeitung mit `busy = TRUE` an.
- Danach wird ein Fehler bei der Bausteinbearbeitung festgestellt.
- Dieser Fehler wird über `error = TRUE` und einer entsprechenden ID am Ausgang `errorId` angezeigt.
- Mit `execute = TRUE` kann der Baustein neu gestartet werden.

3.2.5 Fehlermeldungen

Tabelle 3-7

Fehlernummer [HEX]	Bedeutung
16#0000	Kein Fehler
16#1005	Interner Fehler: Der interne Schrittmerker des FBs hat einen ungültigen Wert
16#1006	Interner Fehler: Die Laufzeit der Funktion `_getStateOfAllDpStations()` hat die maximale Zeit von 20 Sekunden überschritten
16#FFFF80xx	Siehe Onlinehilfe zur Systemfunktion `_getStateOfAllDpStations()`

3.3 FBInitNetworkDiagnosticsStructure

Der Funktionsbaustein `FBInitNetworkDiagnosticsStructure` dient zur Initialisierung der Anwenderstruktur `gasMachineNetwork`.

Es werden einmalig mittels der Systemfunktion `_getSegmentIdentification()` die an einer SIMOTION Steuerung projektierten Netzwerke sowie deren Diagnoseadressen ermittelt. Wahlweise können die Diagnoseadressen auch vom Anwender vorgegeben werden.

Für jedes ermittelte Netzwerk werden anschließend alle zugehörigen Adressen (d.h. Diagnoseadressen und E/A-Adressen) mittels der Systemfunktion `_getNextLogAddress()` ermittelt. Die Adressinformationen werden sowohl netzwerk- und stationszugehörig in der Anwenderstruktur `gasMachineNetwork` als auch in der Struktur `gsCounterArray` adresszugehörig hinterlegt. Dies garantiert später eine performante Aktualisierung der Daten in der Anwenderstruktur `gasMachineNetwork`.

Anschließend werden mittels des Funktionsbausteins `FBGetStateOfAllStationsAtNetwork` die Zustände der sich im Netzwerk befindlichen Stationen ermittelt und ebenfalls in der Anwenderstruktur `gasMachineNetwork` hinterlegt.

Die Abarbeitung des Funktionsbausteins ist danach abgeschlossen und kann bei Bedarf vom Anwender erneut angestoßen werden.

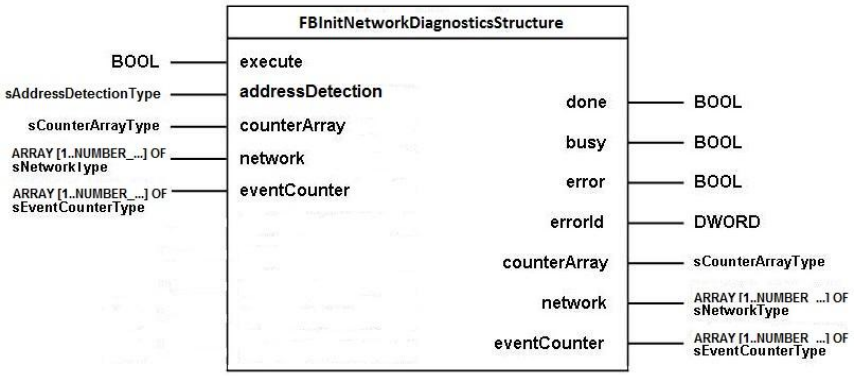
3.3.1 Funktionalität

- Der Baustein ist für den zyklischen Betrieb projektiert.
- Da es keine zeitkritische Anwendung ist, sollte die `BackgroundTask` dafür bevorzugt genutzt werden.
- Mit einer positive Flanke am Eingang `execute` wird die Funktionalität gestartet.
- Über den Eingang `addressDetection` des Funktionsbausteins kann der Anwender festlegen, ob die Diagnoseadressen der an der SIMOTION Steuerung projektierten Netzwerke manuell vorgegeben oder automatisch ermittelt werden sollen.
- Der Ausgang `busy` zeigt die Abarbeitung des Bausteins an.
- Nach Beendigung des Bausteins zeigt der Ausgang `done = TRUE` an, dass gültige Daten vorliegen.
- Tritt während der Abarbeitung des Bausteins ein Fehler auf, so wird dies am Ausgang `error` angezeigt. Es kann dann mit der dazu ausgegebenen Error-ID am Ausgang `errorId` die Fehlerursache nachvollzogen werden.
- Es werden intern die Systemfunktionen `_getSegmentIdentification()`, `_getNextLogAddress()` sowie `_getStateOfAllDpStations()` in Form des Funktionsbausteins `FBGetStateOfAllStationsAtNetwork` aufgerufen.
- Die Adressinformationen der vom Funktionsbaustein ermittelten Adressen (d.h. in welchem Netzwerk und welcher Station sich die jeweilige Adresse befindet) werden in der Struktur `gsCounterArray` hinterlegt. Diese Struktur besteht aus einem Array der Größe `[0..16383]`.

- Hierbei werden z.B. die Informationen der Adresse 256 im Arrayelement 256, die Informationen der Adresse 16383 im Arrayelement 16383 abgelegt. Dieses Vorgehen garantiert später eine performante Aktualisierung der Daten der Anwenderstruktur `gasMachineNetwork`.
  Jeweils die erste Adresse einer neu ermittelten Station wird als Diagnoseadresse und alle weiteren als „Moduladressen“ der Station in der Anwenderstruktur `gasMachineNetwork` abgelegt.

3.3.2 Schematische KOP Darstellung

Abbildung 3-3[6]



3.3.3 Parameter

Tabelle 3-8

Element	P-Typ[1)]	Datentyp	M/O[2)]	Initialwert	Bedeutung
execute	IN	BOOL	M	FALSE	TRUE: Start der Funktionalität
address Detection	IN	sAddress Detection Type	M	-	Übergabestruktur zur Festlegung der automatischen Ermittlung der Netzwerk-Diagnoseadressen oder zur manuellen Vorgabe (siehe Kapitel 3.3.6)
counterArray	IN/OUT	sCounter ArrayType	M	-	Übergabestruktur zur Speicherung aller Adressinformationen (siehe Kapitel 3.3.7)
network	IN/OUT	ARRAY [1..NUMBER_...] OF sNetwork Type	M	-	Übergabestruktur zur Speicherung der ermittelten Daten in der Anwenderstruktur (siehe auch Tabelle 3-1)
eventCounter	IN/OUT	ARRAY [1..NUMBER_...] OF sEvent Counter Type	M	-	Übergabestruktur zum Löschen der Fehlerzähler der Stationen bei erneutem Anstoßen der Bausteinfunktionalität (siehe Kapitel 3.3.8)

[6] ARRAY [1..NUMBER_OF_NETWORKS] OF sNetworkType / sEventCounterType

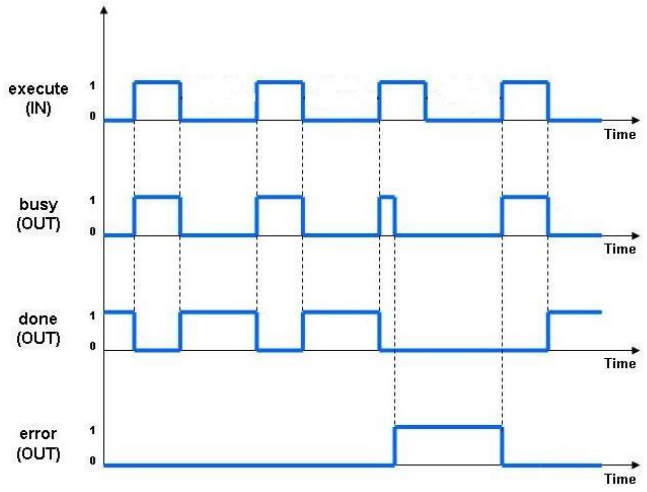
Element	P-Typ[1)]	Datentyp	M/O[2)]	Initialwert	Bedeutung
done	OUT	BOOL	-	FALSE	TRUE: Funktionsbaustein in Bearbeitung
busy	OUT	BOOL	-	FALSE	TRUE: Funktionsbaustein erfolgreich beendet
error	OUT	BOOL	-	FALSE	TRUE: Funktionsbaustein mit Fehler abgebrochen
errorId	OUT	DWORD	-	16#00_00_00_00	Fehlercode

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

3.3.4 Zeitablaufdiagramm

Abbildung 3-4



Bausteinabarbeitung ohne Fehler

- Der Baustein wird mit einer steigenden Flanke am Eingang `execute` gestartet.
- Mit dem Start zeigt der Baustein seine aktive Bearbeitung mit `busy = TRUE` an.
- Wenn die Bausteinbearbeitung wieder abgeschlossen ist, wird der Ausgang `busy = FALSE`.
- Gleichzeitig meldet der Baustein über `done = TRUE`, dass die Bausteinfunktionalität erfolgreich abgeschlossen wurde.
- Wird `execute` vor Beendigung der Funktionalität erneut von FALSE auf TRUE gesetzt, wird die aktuelle Bearbeitung abgebrochen und der Baustein beginnt wieder mit der Abarbeitung der Funktionalität.
- Es werden alle Ausgänge initialisiert.

Bausteinabarbeitung mit Fehler

- Der Baustein wird mit einer steigenden Flanke an `execute` gestartet.
- Mit dem Start zeigt der Baustein seine aktive Bearbeitung mit `busy = TRUE` an.
- Danach wird ein Fehler bei der Bausteinbearbeitung festgestellt.
- Dieser Fehler wird über `error = TRUE` und einer entsprechenden ID am Ausgang `errorId` angezeigt.
- Mit `execute = TRUE` kann der Baustein neu gestartet werden.

3.3.5 Fehlermeldungen

Tabelle 3-9

Fehlernummer [HEX]	Bedeutung
16#0000	Kein Fehler
16#1007	Interner Fehler: Der interne Schrittmerker des FBs hat einen ungültigen Wert
16#FFFF80xx	Siehe Onlinehilfen zu Systemfunktionen • `_getSegmentIdentification()` • `_getNextLogAddress()` • `_getStateOfAllDpStations()`

3.3.6 Übergabestruktur „addressDetection“

Jedes projektierte Netzwerk einer SIMOTION Steuerung besitzt eine eigene eindeutige Diagnoseadresse, die zur Diagnose der zugehörigen Stationen vom Funktionsbaustein `FBInitNetworkDiagnosticsStructure` verwendet wird.

Für den Anwender besteht die Möglichkeit diese Diagnoseadressen vom Baustein automatisch ermitteln zu lassen oder manuell am Baustein vorzugeben.

Die jeweiligen Einstellungen hierfür werden an der Struktur `gsNetworkAddressDetectionSettings` vom Typ `sAddressDetectionSettings` vorgenommen, die an den Eingang `addressDetection` am Baustein übergeben wird.

Struktur `sAddressDetectionSettings`

Tabelle 3-10[7]

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sAddress Detection Settings						
	boAuto DetectionOf Network Addresses	BOOL	IN	M	TRUE	Netzwerk-Diagnoseadressen werden automatisch ermittelt

[7] ARRAY [1..NUMBER_OF_NETWORKS] OF UDINT

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sAddress Detection Settings						
	boManual Network Address Presetting	BOOL	IN	M	FALSE	TRUE: Netzwerk-Diagnoseadressen müssen manuell vorgegeben werden
	au32 Network Diagnostic Address	ARRAY [1..NUMBER_...] OF UDINT	IN	O	0	Array, dessen Elemente mit den gewünschten Netzwerk-Diagnoseadressen vorbelegt werden können; Größe des Arrays ist abhängig vom Wert der Konstanten

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

Automatische Ermittlung der Netzwerk-Diagnoseadressen (Defaulteinstellung)

Sollen die Netzwerk-Diagnoseadressen automatisch vom Funktionsbaustein ermittelt werden, so ist die Variable `boAutoDetectionOfNetworkAddresses` auf den Wert TRUE zu setzen. Dies passiert einmalig automatisch beim ersten Aufruf des Bausteins nach Download der SIMOTION Steuerung. Zeitgleich wird die Variable `boManualNetworkAddressPresetting` auf den Wert FALSE gesetzt, falls diese vorher den Wert TRUE hatte (siehe auch Kapitel 5.2).

Die Systemfunktion `_getSegmentIdentification()`, die bausteinintern aufgerufen wird, liefert in diesem Fall alle Netzwerk-Diagnoseadressen.

Beispiel

- HW-Konfiguration der SIMOTION Steuerung
  - PROFIBUS Netzwerk an PROFIBUS Schnittstelle X136 (Diagnoseadresse **16382**)
  - PROFIBUS_Integrated (Diagnoseadresse **16381**)
  - PROFINET Netzwerk an PROFINET Schnittstelle X150 (Diagnoseadresse **16349**)
- Die Konstante `NUMBER_OF_NETWORKS` hat einen Wert größer gleich 3
- Die Variable `boAutoDetectionOfNetworkAddresses` hat den Wert TRUE
- Über die Variable `gboActualisationOfNetworkDiagnosticsStructure` kann die Bausteinfunktionalität erneut angestoßen werden, wenn diese auf den Wert TRUE gesteuert wird

- Die Netzwerk-Diagnoseadressen werden bausteinintern von der Systemfunktion `_getSegmentIdentification()` ermittelt
- Der Funktionsbaustein ermittelt alle netzwerkzugehörigen Adressen (d.h. Diagnoseadressen sowie E/A-Adressen der projektierten Teilnehmer) sowie die aktuellen Zustände der Stationen und hinterlegt die Daten in der Anwenderstruktur `gasMachineNetwork`.
- Somit würde die Anwenderstruktur wie folgt aussehen:
  - `gasMachineNetwork[1]` enthält die Daten für das PROFIBUS Netzwerk (X136) mit der Diagnoseadresse **16382**
  - `gasMachineNetwork[2]` enthält die Daten für den PROFIBUS_Integrated mit der Diagnoseadresse **16381**
  - `gasMachineNetwork[3]` enthält die Daten für das PROFINET Netzwerk (X1400) mit der Diagnoseadresse **16349**
- **Vorteil:**
  - Die Netzwerk-Diagnoseadressen werden automatisch vom Baustein ermittelt, somit müssen diese dem Anwender nicht bekannt sein, um die Netzwerke seiner SIMOTION Steuerung diagnostizieren zu können.
  - Die Netzwerk-Diagnoseadressen müssen im Gegensatz zur manuellen Vorgabe am Baustein nicht abgeändert werden, falls sich diese in der HW-Konfiguration ändern.
- **Nachteil:**
  - Der Anwender hat keinen Einfluss darauf, welche Netzwerke seiner SIMOTION Steuerung diagnostiziert werden. Es werden grundsätzlich alle Netzwerke diagnostiziert, sofern der Wert der Konstanten `NUMBER_OF_NETWORKS` größer gleich der projektierten Anzahl an Netzwerken entspricht.

Manuelle Vorgabe der Netzwerk-Diagnoseadressen

Sollen die Netzwerk-Diagnoseadressen manuell vom Anwender am Funktionsbaustein vorgegeben werden, so ist die Variable `boManualNetworkAddressPresetting` auf den Wert `TRUE` zu setzen. Zeitgleich wird die Variable `boAutoDetectionOfNetworkAddresses` auf den Wert `FALSE` gesetzt, falls diese vorher den Wert `TRUE` hatte (siehe auch Kapitel 5.4)

Die Elemente des Arrays `au32NetworkDiagnosticAddress` müssen dann mit den Netzwerk-Diagnoseadressen vorbelegt werden, deren Stationen vom Funktionsbaustein diagnostiziert werden sollen.

Beispiel

- HW-Konfiguration der SIMOTION Steuerung
  - PROFIBUS Netzwerk an PROFIBUS Schnittstelle X136 (Diagnoseadresse **16382**)
  - PROFIBUS_Integrated (Diagnoseadresse **16381**)
  - PROFINET Netzwerk an PROFINET Schnittstelle X150 (Diagnoseadresse **16349**)
- Die Variable `boManualNetworkAddressPresetting` hat den Wert `TRUE`

- Der Anwender möchte nur das PROFIBUS Netzwerk an der Schnittstelle X136 sowie das PROFINET Netzwerk an der PROFINET Schnittstelle X150 diagnostizieren
- Die Konstante `NUMBER_OF_NETWORKS` hat einen Wert größer gleich 2
- Die Elemente des Arrays `au32NetworkDiagnosticAddress` werden wie folgt vorbelegt:
  - `au32NetworkDiagnosticAddress[1] = 16382`
  - `au32NetworkDiagnosticAddress[2] = 16349`
  - `au32NetworkDiagnosticAddress[x] = 0`
- Über die Variable `gboActualisationOfNetworkDiagnosticsStructure` kann die Bausteinfunktionalität erneut angestoßen werden, wenn diese auf den Wert `TRUE` gesteuert wird
- Der Funktionsbaustein ermittelt alle netzwerkzugehörigen Adressen (d.h. Diagnoseadressen sowie E/A-Adressen der projektierten Teilnehmer) sowie die aktuellen Zustände der Stationen und hinterlegt die Daten in der Anwenderstruktur `gasMachineNetwork`.
- Somit würde die Anwenderstruktur wie folgt aussehen:
  - `gasMachineNetwork[1]` enthält die Daten für das PROFIBUS Netzwerk (X136) mit der Diagnoseadresse **16382**
  - `gasMachineNetwork[2]` enthält die Daten für das PROFINET Netzwerk (X1400) mit der Diagnoseadresse **16349**
  - `gasMachineNetwork[x]` enthält keine Daten
- **Vorteil:**
  - Der Anwender kann bestimmen, welche und wie viele Netzwerke seiner SIMOTION Steuerung diagnostiziert werden sollen, indem nur die gewünschten Netzwerk-Diagnoseadressen am Funktionsbaustein vorgegeben werden.
- **Nachteil:**
  - Dem Anwender müssen die entsprechenden Netzwerk-Diagnoseadressen bekannt sein, um die gewünschten Netzwerke seiner SIMOTION Steuerung diagnostizieren zu können.
  - Falls sich die Diagnoseadressen in der HW-Konfiguration ändern, so müssen diese am Baustein ebenfalls abgeändert werden.

3.3.7 Übergabestruktur „counterArray“

Die Struktur `gsCounterArray` vom Typ `sCounterArrayType` dient zur Speicherung aller Adressinformationen der Adressen, die vom Funktionsbaustein für die jeweiligen Netzwerke der SIMOTION Steuerung ermittelt wurden.
Diese Struktur wird an den Bausteineingang `counterArray` übergeben.

Die Struktur setzt sich aus dem Array `asNetworkIndexAndStationNumber` zusammen, das aus insgesamt `16384` Arrayelementen besteht (von `0` bis `16383`, wobei `16383` die größtmögliche Diagnoseadresse im SIMOTION Adressraum darstellt).

Jedes Element entspricht dabei einer möglichen Adresse (d.h. Diagnose- oder E/A-Adresse), die in der HW-Konfiguration vorkommen kann.
Die Adressinformationen für die jeweilige Adresse werden im korrespondierenden Arrayelement hinterlegt.

Struktur `sCounterArrayType`

Tabelle 3-11

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sCounter ArrayType						
	asNetwork IndexAnd Station Number	ARRAY [0..16383] OF sAddressType	OUT	---	---	Array zur Speicherung der zusätzlichen Adressinformation zur jeweiligen Adresse

Struktur `sAddressType`

Tabelle 3-12

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sAddress Type						
	sInput	sCounterType	OUT	---	---	Struktur für Eingangsadresse
	sOutput	sCounterType	OUT	---	---	Struktur für Ausgangsadresse
	sDiagnostic	sCounterType	OUT	---	---	Struktur für Diagnoseadresse

Struktur `sCounterType`

Tabelle 3-13

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sCounter Type						
	u8Network Index	USINT	OUT	---	0	Nummer des Netzwerks, in dem die jeweilige Adresse ermittelt wurde
	u8Station Number	USINT	OUT	---	0	Nummer der Station, an der die jeweilige Adresse projektiert wurde

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

Beispiel

- In der HW-Konfiguration wurden folgende Adressen projektiert:
  - Diagnoseadresse **16361**
  - Eingangsadresse **292**
  - Ausgangsadresse **1024**
- Die vom Baustein zusätzlich ermittelten Adressinformationen werden in der Übergabestruktur `counterArray` in folgenden Arrayelementen hinterlegt:
  - `asNetworkIndexAndStationNumber[16361]`
  - `asNetworkIndexAndStationNumber[292]`
  - `asNetworkIndexAndStationNumber[1024]`

Da eine Adresse mehrmals vorkommen kann (z.B. Eingangsadresse 292 und Ausgangsadresse 292), wird beim Hinterlegen der Adressinformationen in den Arrayelementen nochmals zwischen Eingangs-, Ausgangs- und Diagnoseadresse unterschieden.

Die Adressinformation einer Adresse enthält neben der Nummer des Netzwerks auch die Stationsnummer, in der die Adresse vom Baustein ermittelt wurde.

Diese Informationen werden später in der `PeripheralFaultTask` verwendet, um eine performante Aktualisierung der Anwenderstruktur `gasMachineNetwork` zu ermöglichen.

3.3.8 Übergabestruktur „eventCounter“

Die Struktur `gasEventCounter`, die aus einem Array vom Typ `sEventCounterType` besteht, dient an diesem Baustein lediglich zum Zurücksetzen der Fehlerzähler aller Station im jeweiligen Netzwerk, wenn die Bausteinfunktionalität vom Anwender erneut angestoßen wird.
Die Struktur wird an den Bausteineingang `eventCounter` übergeben.

Struktur `sEventCounterType`

Tabelle 3-14[8]

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sEvent Counter Type						
	asStation	ARRAY [1..HIGHEST_...] OF sStationType	OUT	---	---	Array der Struktur, die für alle Stationen pro Netzwerk Fehlerzähler enthält

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

[8] ARRAY [1..HIGHEST_STATION_ADDRESS_AT_NETWORK] OF sStationType

Struktur `sStationType`

Tabelle 3-15

Struktur	Parameter	Datentyp	P-Typ[1)]	M/O[2)]	Initialwert	Beschreibung
sStation Type						
	u16Last EventIndex	UINT	OUT	---	0	Zähler für Ringpuffer der jeweiligen Station; enthält den Index des Elements, in das zuletzt ein Fehler eingetragen wurde
	u16Number OfEvents	UINT	OUT	---	0	Anzahl an Fehlern, die insgesamt an der jeweiligen Station anstehen
	au16Events OnModule	ARRAY [1..HIGHEST_...] OF UINT	OUT	---	---	Anzahl an Fehlern, die insgesamt für das jeweilige Modul der Station anstehen

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

Die Struktur `gasEventCounter` wird zur Laufzeit in der `PeripheralFaultTask` aktualisiert. Hierbei werden die sogenannten Task-Start-Informationen (TSI) ausgewertet und entsprechend die Fehlerzähler der fehlerhaften Stationen erhöht bzw. verringert.

So ist es möglich mehrere Fehler einer Station bzw. mehrere Fehler an einem Modul zu erfassen und entsprechend den aktuellen Status der Station in der Anwenderstruktur `gasMachineNetwork` anzupassen.

Beispiel

- An einer Station treten mehrere Fehler gleichzeitig oder nacheinander auf.
- Die Fehlerzähler der jeweiligen Module der Station werden um die Anzahl der Fehler erhöht, die an diesem Modul aufgetreten sind. Zudem wird eine Gesamtanzahl an Fehlern errechnet, die an der Station anstehen.
- Der aktuelle Zustand der Station wird in der Anwenderstruktur `gasMachineNetwork` als `DISTURBED` gemeldet.
- Es wird nun einer der anstehenden Fehler an der Station behoben. Der Fehlerzähler des entsprechenden Moduls wird um den Wert `1` verringert, sowie die Gesamtanzahl an Fehlern angepasst.
- Der aktuelle Zustand der Station wird in der Anwenderstruktur immer noch als `DISTURBED` gemeldet.
- Erst wenn alle Fehler an der Station behoben wurden, so wird die Station in der Anwenderstruktur wieder als `IN_OPERATION` gemeldet.

3.4 FCMachinePeripheralFailureHandling

Die Funktion `FCMachinePeripheralFailureHandling` dient zur Auswertung der Task-Start-Informationen (TSI) der `PeripheralFaultTask` und zur Aktualisierung der Anwenderstruktur `gasMachineNetwork`.

Es werden sowohl kommende als auch gehende Fehler ausgewertet und die Fehlerinformationen entsprechend in der Anwenderstruktur netzwerk- und stationszugehörig eingetragen.
Zudem werden die Fehlerzähler der fehlerhaften Stationen in der Struktur `gasEventCounter` inkrementiert bzw. dekrementiert, wodurch die aktuellen Zustände der Stationen in der Anwenderstruktur entsprechend angepasst werden.

3.4.1 Funktionalität

- Die Funktion muss in der `PeripheralFaultTask` aufgerufen werden.
- Es werden sowohl kommende als auch gehende Fehler ausgewertet und mittels der in der Struktur `gasCounterArray` hinterlegten Adressinformationen in der Anwenderstruktur netzwerk- und stationszugehörig hinterlegt.
- Abhängig davon, ob Fehler an einer Station anstehen oder die Station ausgefallen ist wird der aktuelle Status der Station in der Anwenderstruktur aktualisiert.
- Es werden neben Ausfall und Wiederkehr einer Station auch Fehler an deren Modulen erkannt (z.B. Ziehen / Stecken, Drahtbruch / Kurzschluss, etc.).
- Folgende Task-Start-Informationen (TSI) werden fehlerabhängig ausgewertet:
  - **TSI#startTime** → Startzeitpunkt der `PeripheralFaultTask`
  - **TSI#interruptId** → gemeldete Fehler-ID
  - **TSI#logDiagAdr** → Diagnoseadresse der fehlerhaften Station
  - **TSI#logBaseAdrIn** → Eingangsadresse der fehlerhaften Station
  - **TSI#logBaseAdrOut** → Ausgangsadresse der fehlerhaften Station
  - **TSI#eventClass** → Ereignisklasse (abhängig von `TSI#interruptId`) als weitere Fehlerinformation
  - **TSI#faultId** → Störungskennung (abhängig von `TSI#interruptId` und `TSI#eventClass`) als weitere Fehlerinformation
  - **TSI#details** → Detailinformationen (abhängig von `TSI#interruptId` und `TSI#eventClass`) als weitere Fehlerinformation

Übersicht der Fehler-IDs, die ausgewertet werden

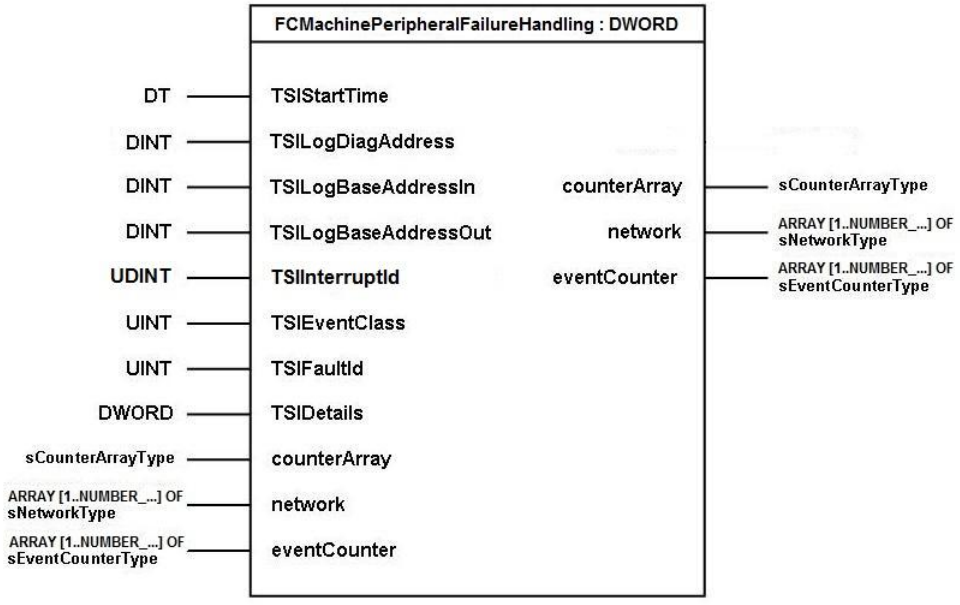
Tabelle 3-16

TSI#interruptId	Nummer	Bedeutung
_SC_PROCESS_INTERRUPT	200	Prozessalarm an Peripheriebaugruppe aufgetreten
_SC_DIAGNOSTIC_INTERRUPT	201	Diagnosealarm an Peripheriebaugruppe aufgetreten
_SC_STATION_DISCONNECTED	202	Stationsausfall eines IO-Device / DP-Slave
_SC_STATION_RECONNECTED	203	Stationswiederkehr eines IO-Device / DP-Slave
_SC_IMAGE_UPDATE_FAILED	204	Fehler bei Bilden des Prozessabbilds
_SC_IMAGE_UPDATE_OK	206	Bilden des Prozessabbilds funktioniert wieder
_SC_IO_MODULE_SYNCHRONIZED	214	Synchronisation eines Moduls erreicht
_SC_IO_MODULE_NOT_SYNCHRONIZED	215	Synchronisation eines Moduls ausgefallen
_SC_PULL_PLUG_INTERRUPT	216	Ziehen oder Stecken von Baugruppen (Modulen) eines IO-Device / DP-Slave

Hinweis Weitere Informationen zum Thema „Task-Start-Informationen (TSI)“ finden Sie in der SIMOTION SCOUT Onlinehilfe unter dem Stichwort „TSI“.

3.4.2 Schematische KOP Darstellung

Abbildung 3-5[9]



[9] ARRAY [1..NUMBER_OF_NETWORKS] OF sNetworkType / sEventCounterType

3.4.3 Parameter

Tabelle 3-17[10]

Element	P-Typ[1)]	Datentyp	M/O[2)]	Initialwert	Bedeutung
TSIStartTime	IN	DT	M	DT#0001-01-01-0:0:0	Startzeitpunkt der `PeripheralFaultTask`
TSILogDiag Address	IN	DINT	M	0	Diagnoseadresse der fehlerhaften Station
TSILogBase AddressIn	IN	DINT	M	0	Eingangsadresse der fehlerhaften Station
TSILogBase AddressOut	IN	DINT	M	0	Ausgangsadresse der fehlerhaften Station
TSIInterruptId	IN	UDINT	M	0	gemeldete Fehler-ID
TSIEventClass	IN	UINT	M	16#00_00	Ereignisklasse (abhängig von `TSIInterruptId`) als weitere Fehlerinformation
TSIFaultId	IN	UINT	M	16#00_00	Störungskennung (abhängig von `TSIInterruptId` und `TSIEventClass`) als weitere Fehlerinformation
TSIDetails	IN	DWORD	M	16#00_00_00_00	Detailinformationen (abhängig von `TSIInterruptId` und `TSIEventClass`) als weitere Fehlerinformation
counterArray	IN/OUT	sCounterArray Type	M	---	Übergabestruktur für die bereits ermittelten Adressinformationen
network	IN/OUT	ARRAY [1..NUMBER_...] OF sNetworkType	M	---	Struktur zur Übergabe der Anwenderstruktur
eventCounter	IN/OUT	ARRAY [1..NUMBER_...] OF sEventCounter Type	M	---	Struktur zur Übergabe der Fehlerzähler der Stationen

1) Parametertypen:
IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

2) Parameterart:
M = Pflichtparameter, O = Optionaler Parameter

[10] ARRAY [1..NUMBER_OF_NETWORKS] OF sNetworkType / sEventCounterType

3.4.4 Fehlermeldungen

Tabelle 3-18

Fehlernummer [HEX]	Bedeutung
16#0000	Kein Fehler
16#1008	Für die gemeldete fehlerhafte Adresse konnte kein Netzwerkindex und / oder keine Stationsnummer in den Adressinformationen gefunden werden
16#1009	Die von der `PeripheralFaultTask` gemeldete `FaultId` ist unbekannt
16#1010	Die von der `PeripheralFaultTask` gemeldete Diagnoseadresse ist unbekannt
16#1011	Die von der `PeripheralFaultTask` gemeldete logische Adresse ist unbekannt

4 Inbetriebnahme der Applikation

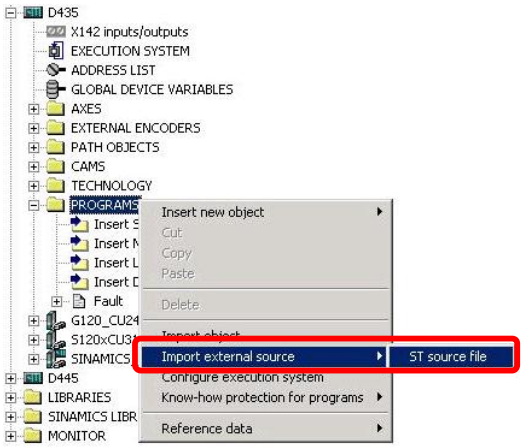
4.1 Übersicht

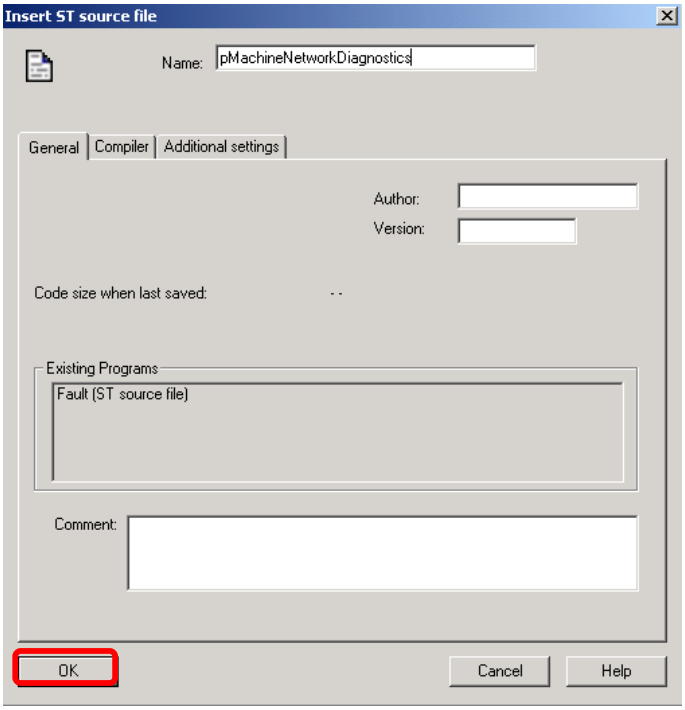
- Das mit diesem Applikationsbeispiel mitgelieferte ZIP-Archiv `53705461_SIMOTION_Diagnostics_V2_0.zip` enthält die zur Diagnose benötigten ST-Quellen.
- In der ST-Quelle `fMachineNetworkDiagnostics` sind die Funktionsbausteine `FBGetStateOfAllStationsAtNetwork` und `FBInitNetworkDiagnosticsStructure` sowie die Funktion `FCMachinePeripheralFailureHandling` enthalten.
- In der ST-Quelle `pMachineNetworkDiagnostics` sind die Programme für die `BackgroundTask` sowie die `PeripheralFaultTask` enthalten, in denen die Funktionsbausteine sowie die Funktion aufgerufen werden.
- Das Programm `pMachineNetworkDiagnosticsBackgroundTask` sollte bevorzugt in der `BackgroundTask` eingebunden werden.
  Das Programm `pMachineNetworkDiagnosticsPeripheralFaultTask` muss in der `PeripheralFaultTask` eingebunden werden.

4.2 Einbinden der ST-Quellen

Die ST-Quellen `fMachineNetworkDiagnostics` sowie `pMachineNetworkDiagnostics` können sowohl in ein neues als auch in ein bestehendes Projekt eingebunden werden.

Tabelle 4-1

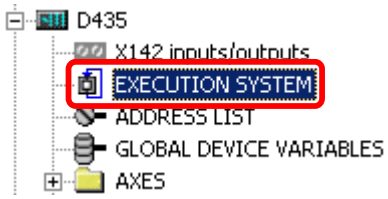
Nr.	Aktion
1.	Extrahieren Sie das mitgelieferte ZIP-Archiv.
2.	Öffnen Sie das Projekt, in das die ST-Quellen eingebunden werden sollen.
3.	Öffnen Sie im Projektbaum die Eigenschaften der gewünschten SIMOTION Steuerung und markieren Sie den Unterordner PROGRAMME. Durch Rechtsklick öffnet sich das Kontextmenü. Wählen Sie hier die Option **„Externe Quelle importieren > ST-Quelle“**. 

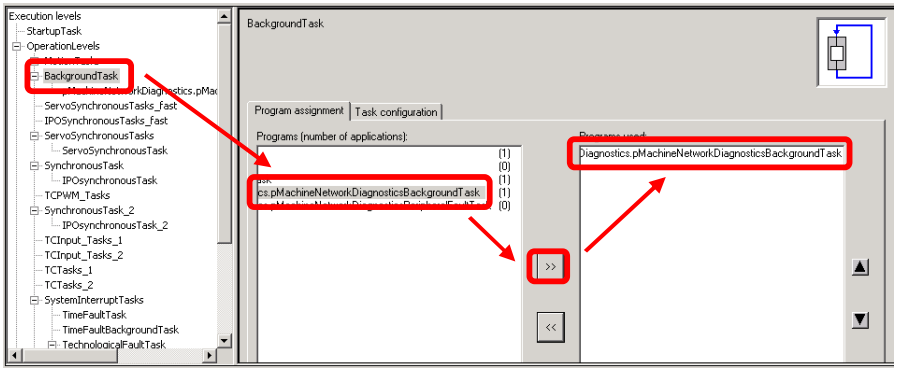
Nr.	Aktion
4.	Markieren Sie die ST-Quellen `fMachineNetworkDiagnostics` und `pMachineNetworkDiagnostics` (Taste `Strg` gedrückt halten) und klicken Sie auf die Schaltfläche **„Öffnen“**.
5.	Starten Sie das Importieren mittels der Schaltfläche **„OK“**. Es müssen keine weiteren Einstellungen vorgenommen werden. 
6.	Speichern und übersetzen Sie das Projekt. 

4.3 Einbinden der Programme

Nach dem Importieren der ST-Quellen müssen die Programme in die entsprechenden Tasks eingebunden werden.

Tabelle 4-2

Nr.	Aktion
1.	Öffnen Sie im Projektbaum das `Ablaufsystem` der SIMOTION Steuerung. 

Nr.	Aktion
2.	Markieren Sie die `BackgroundTask` und binden Sie das Programm `pMachineNetworkDiagnosticsBackgroundTask` über die Schaltfläche **„>>“** in die Task ein.  **Hinweis** Gehen Sie analog für das Programm `pMachineNetworkDiagnosticsPeripheralFaultTask` vor. Achten Sie darauf, das das Program in der `PeripheralFaultTask` eingebunden werden muss!
3.	Speichern und Übersetzen Sie das Projekt. 
4.	Laden Sie die geänderten Einstellungen in die SIMOTION Steuerung. 

5 Bedienung der Applikation

5.1 Werte der globalen Konstanten anpassen

Im Applikationsbeispiel werden globale Konstanten verwendet, die vom Anwender angepasst werden können, um die Diagnosefunktionalität weiter zu optimieren.

Tabelle 5-1

Nr.	Aktion

1.	Die globalen Konstanten befinden sich in der ST-Quelle `fMachineNetworkDiagnostics` und sind mit Standardwerten vorbelegt (siehe auch Kapitel 2.2: Beschreibung der Kernfunktionalität). Passen Sie die Werte entsprechend Ihrer Konfiguration an.
	
	`VAR_GLOBAL CONSTANT`
	`  NUMBER_OF_NETWORKS : USINT := 5;`
	`  HIGHEST_STATION_ADDRESS_AT_NETWORK : USINT := 128;`
	`  HIGHEST_NUMBER_OF_MODULES_PER_STATION : USINT := 64;`
	`  NUMBER_OF_STORED_EVENTS_PER_STATION : USINT := 8;`
	`END_VAR;`

2.	Speichern und Übersetzen Sie das Projekt. 
3.	Laden Sie die geänderten Einstellungen in die SIMOTION Steuerung. 

Hinweis Die Werte der Konstanten sollten immer entsprechend der SIMOTION Konfiguration gewählt werden, da sonst jeweils einmalig nach Anstoßen der Diagnosefunktionalität mehr Zykluszeit in der `BackgroundTask` verbraucht wird als unbedingt notwendig.

Hinweis Soll von einem HMI aus auf die Variablen der Diagnosefunktionalität zugegriffen werden (d.h. auf Variablen der Anwenderstruktur `gasMachineNetwork`), so ist zu beachten, dass alle HMI-relevanten Variablen unterhalb der **64kByte** Adressgrenze für HMI-Zugriffe liegen müssen.

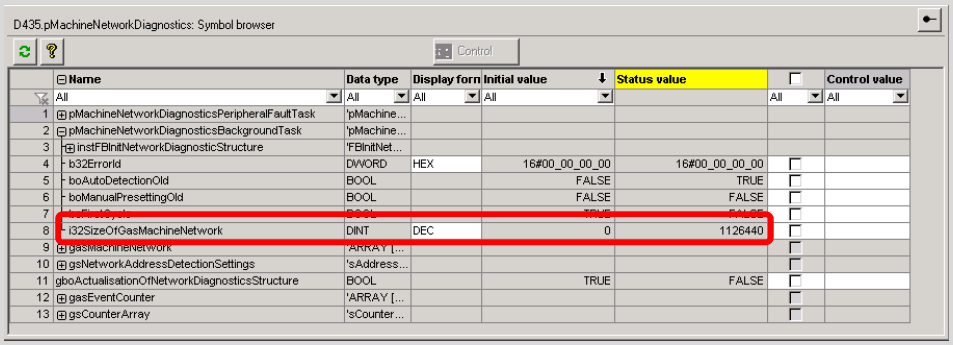
Wird die **64kByte** Adressgrenze überschritten, so wird vom Compiler beim Übersetzen eine Warnung ausgegeben, dass ab der Anwenderstruktur `gasMachineNetwork` ein HMI-Zugriff auf Variablen nicht mehr möglich ist:

Level	Message
Information	START of compilation of 'pMachineNetworkDiagnostics' at 15:35:37
Warning	pMachineNetworkDiagnostics(27) : 32050 :Variables as of "gasMachineNetwork" are not visible for the HMI
Information	END of compilation of 'pMachineNetworkDiagnostics' at 15:35:37
Information	Compilation of pMachineNetworkDiagnostics: 0 Error, 1 Warning(s)

Bei Verwendung der Standardwerte der Konstanten ergibt sich für die Anwenderstruktur `gasMachineNetwork` eine Größe von ca. **1125kByte**, die darin enthaltenen Daten sind somit vom HMI aus nicht mehr erreichbar.
Durch Anpassen der Werte kann die Größe der Anwenderstruktur entsprechend verkleinert werden.

Hinweis

Die aktuelle Größe der Anwenderstruktur wird in der Variable `i32SizeOfGasMachineNetwork` angezeigt. Diese wird im Programm `pMachineNetworkDiagnosticsBackgroundTask` jeweils einmalig nach Download der SIMOTION Steuerung beschrieben.



	Name	Data type	Display form	Initial value	Status value		Control value
	All	All	All	All		All	All
1	pMachineNetworkDiagnosticsPeripheralFaultTask	'pMachine...					
2	pMachineNetworkDiagnosticsBackgroundTask	'pMachine...					
3	instFBInitNetworkDiagnosticStructure	'FBInitNet...					
4	b32ErrorId	DWORD	HEX	16#00_00_00_00	16#00_00_00_00		
5	boAutoDetectionOld	BOOL		FALSE	TRUE		
6	boManualPresettingOld	BOOL		FALSE	FALSE		
7	[illegible]	[illegible]		[illegible]	[illegible]		
8	i32SizeOfGasMachineNetwork	DINT	DEC	0	1126440		
9	gasMachineNetwork	'ARRAY [...					
10	gsNetworkAddressDetectionSettings	'sAddress...					
11	gboActualisationOfNetworkDiagnosticsStructure	BOOL		TRUE	FALSE		
12	gasEventCounter	'ARRAY [...					
13	gsCounterArray	'sCounter...					

Kann die Größe der Anwenderstruktur aufgrund der SIMOTION Konfiguration durch Anpassen der Werte der Konstanten nicht auf die Adressgrenze von **64kByte** verkleinert werden, so müssen die gewünschten Daten gegebenenfalls in eine weitere Struktur umkopiert werden, auf die wiederum vom HMI aus zugegriffen werden kann.

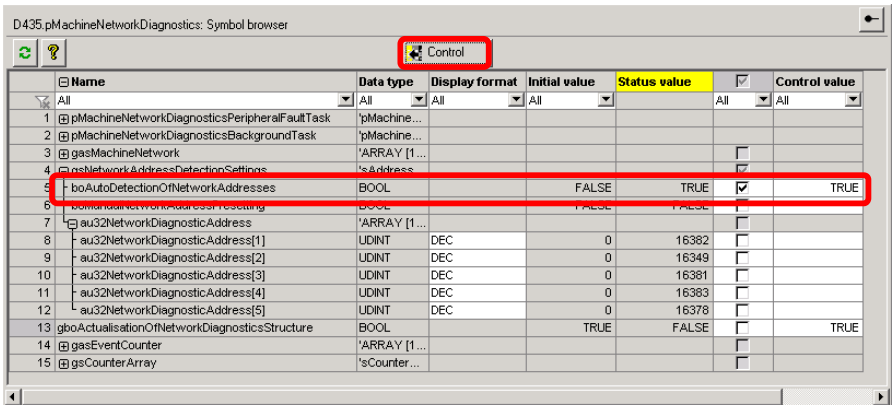
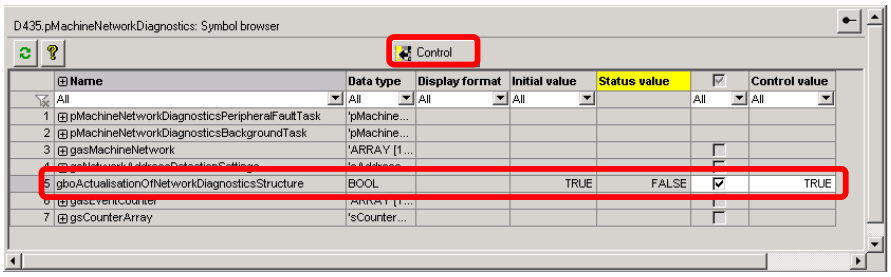
5.2 Automatische Ausführung der Diagnosefunktionalität

Für die automatische Ausführung der Diagnosefunktionalität bedarf es nach Download der ST-Quellen in die SIMOTION Steuerung keinerlei Einstellungen. Sobald die Steuerung in den Betriebszustand **„RUN“** geschaltet wird, werden alle an der SIMOTION Steuerung projektierten Netzwerke automatisch diagnostiziert und die Diagnosedaten in der Anwenderstruktur `gasMachineNetwork` hinterlegt.

Sollte zwischenzeitlich die Funktionalität der manuellen Vorgabe der Netzwerk-Diagnoseadressen verwendet worden sein (siehe Kapitel 5.4), so kann über folgende Einstellung wieder auf die automatische Ermittlung der Diagnosedaten umgestellt werden.

Tabelle 5-2

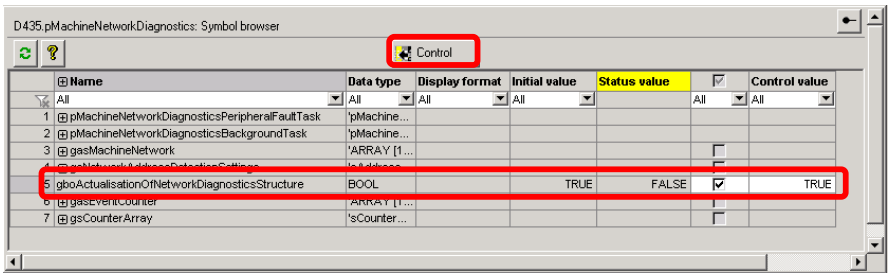
Nr.	Aktion
1.	Markieren Sie hierfür im Ordner PROGRAMME der SIMOTION Steuerung die ST-Quelle `pMachineNetworkDiagnostics`. Im Symbolbrowser wird der Inhalt der ST-Quelle angezeigt. Öffnen Sie dort die globale Struktur `gsNetworkAddressDetectionSettings`.

Nr.	Aktion
2.	Steuern Sie mittels der Schaltfläche **„Steuern“** die Variable `boAutoDetectionOfNetworkAddresses` auf den Wert **„TRUE“**.  D435.pMachineNetworkDiagnostics: Symbol browser Control Name / Data type / Display format / Initial value / Status value / Control value 1 pMachineNetworkDiagnosticsPeripheralFaultTask 'pMachine... 2 pMachineNetworkDiagnosticsBackgroundTask 'pMachine... 3 gasMachineNetwork 'ARRAY [1... 4 gsNetworkAddressDetectionSettings 'sAddress... 5 boAutoDetectionOfNetworkAddresses BOOL FALSE TRUE TRUE 6 boManualNetworkAddressPresetting BOOL FALSE FALSE 7 au32NetworkDiagnosticAddress 'ARRAY [1... 8 au32NetworkDiagnosticAddress[1] UDINT DEC 0 16382 9 au32NetworkDiagnosticAddress[2] UDINT DEC 0 16349 10 au32NetworkDiagnosticAddress[3] UDINT DEC 0 16381 11 au32NetworkDiagnosticAddress[4] UDINT DEC 0 16383 12 au32NetworkDiagnosticAddress[5] UDINT DEC 0 16378 13 gboActualisationOfNetworkDiagnosticsStructure BOOL TRUE FALSE TRUE 14 gasEventCounter 'ARRAY [1... 15 gsCounterArray 'sCounter... **Hinweis** Die Variable `boManualNetworkAddressPresettings` wird automatisch auf den Wert **„FALSE“** gesetzt. Eventuelle im Array `au32NetworkDiagnosticAddress` eingetragene Netzwerk-Diagnoseadressen werden mit dieser Einstellung vom Funktionsbaustein `FBInitNetworkDiagnosticStructure` ignoriert.
3.	Steuern Sie anschließend die globale Variable `gboActualisationOfNetworkDiagnosticsStructure` mittels der Schaltfläche **„Steuern“** auf den Wert **„TRUE“**.  D435.pMachineNetworkDiagnostics: Symbol browser Control Name / Data type / Display format / Initial value / Status value / Control value 1 pMachineNetworkDiagnosticsPeripheralFaultTask 'pMachine... 2 pMachineNetworkDiagnosticsBackgroundTask 'pMachine... 3 gasMachineNetwork 'ARRAY [1... 5 gboActualisationOfNetworkDiagnosticsStructure BOOL TRUE FALSE TRUE 7 gsCounterArray 'sCounter... **Hinweis** Die globale Variable wird automatisch nach Abarbeitung des Funktionsbausteins `FBInitNetworkDiagnosticStructure` auf den Wert **„FALSE“** zurückgesetzt.

5.3 Manuelles Anstoßen der Diagnosefunktionalität

Die Diagnosefunktionalität kann bei Bedarf auch manuell vom Anwender angestoßen werden.

Tabelle 5-3

Nr.	Aktion
1.	Markieren Sie hierfür im Ordner PROGRAMME der SIMOTION Steuerung die ST-Quelle `pMachineNetworkDiagnostics`. Im Symbolbrowser wird der Inhalt der ST-Quelle angezeigt.
2.	Steuern Sie die globale Variable `gboActualisationOfNetworkDiagnosticsStructure` mittels der Schaltfläche **„Steuern“** auf den Wert **„TRUE“**.  **Hinweis** Die globale Variable wird automatisch nach Abarbeitung des Funktionsbausteins `FBInitNetworkDiagnosticStructure` auf den Wert **„FALSE“** zurückgesetzt.
3.	Die Funktionsbausteine `FBInitNetworkDiagnosticStructure` sowie `FBGetStateOfAllStationsAtNetwork` werden erneut durchlaufen. Es werden alle projektierten Stationen sowie deren aktueller Zustand ermittelt und in der Anwenderstruktur `gasMachineNetwork` hinterlegt.

Hinweis

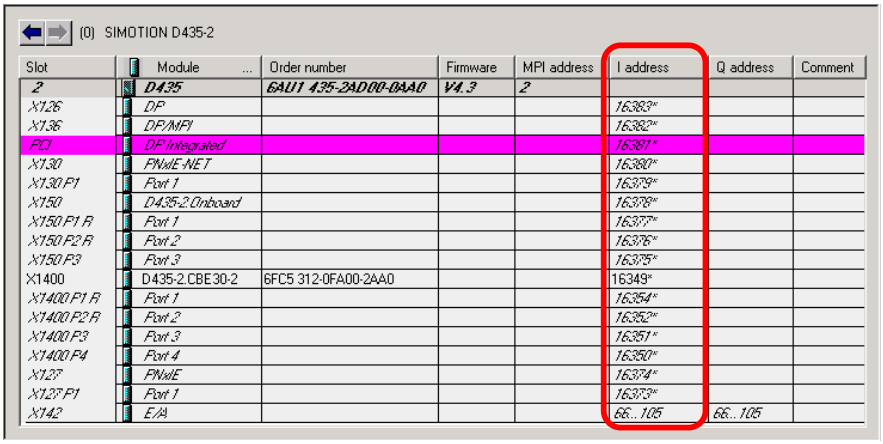
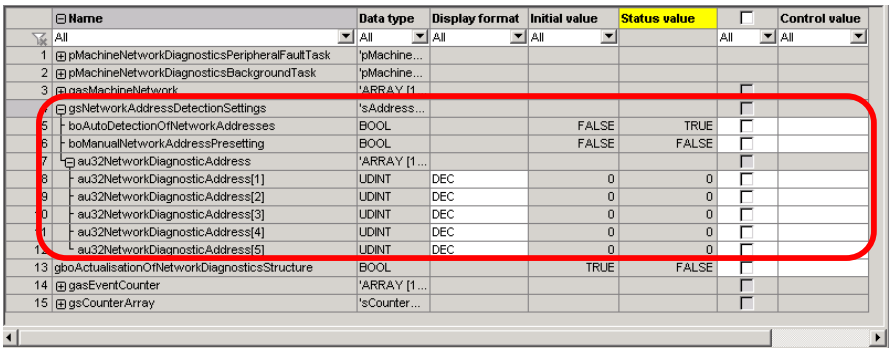
Nach jedem manuellen Anstoßen der Diagnosefunktionalität werden die zu diesem Zeitpunkt in der Anwenderstruktur vorhandenen Daten gelöscht und durch die neuen Daten ersetzt. Bereits eingetragene Fehler an Stationen oder deren Modulen bleiben demnach **nicht** erhalten!

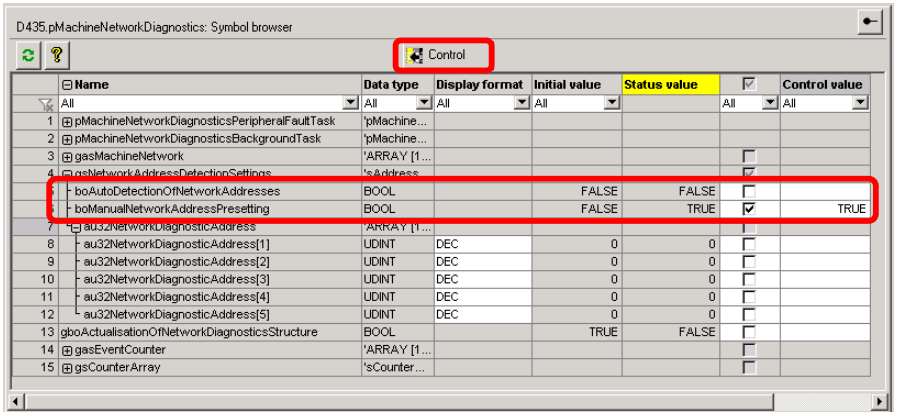
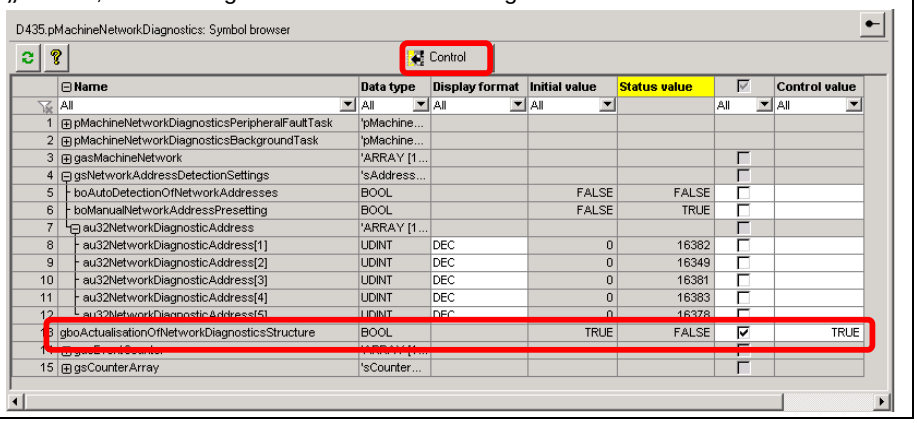
5.4 Manuelle Vorgabe der Netzwerk-Diagnoseadressen

Bei Bedarf (siehe Kapitel 3.3.6: Übergabestruktur „addressDetection“) können die Diagnoseadressen der Netzwerke, die diagnostiziert werden sollen, vom Anwender manuell am Funktionsbaustein `FBInitNetworkDiagnosticStructure` vorgegeben werden.

Tabelle 5-4

Nr.	Aktion
1.	Öffnen Sie die HW-Konfiguration der SIMOTION Steuerung, deren Netzwerke Sie diagnostizieren möchten.

Nr.	Aktion
2.	Markieren Sie die SIMOTION Steuerung. Im Detailfenster am unteren Bildrand erhalten Sie eine Übersicht der Schnittstellen sowie deren Diagnoseadressen.  (0) SIMOTION D435-2 Slot | Module | Order number | Firmware | MPI address | I address | Q address | Comment 2 | D435 | 6AU1 435-2AD00-0AA0 | V4.3 | 2 X126 | DP | 16383* X136 | DP/MPI | 16382* PCI | DP Integrated | 16381* X130 | PN/E-NET | 16380* X130 P1 | Port 1 | 16379* X150 | D435-2 Onboard | 16378* X150 P1 R | Port 1 | 16377* X150 P2 R | Port 2 | 16376* X150 P3 | Port 3 | 16375* X1400 | D435-2 CBE30-2 | 6FC5 312-0FA00-2AA0 | 16349* X1400 P1 R | Port 1 | 16354* X1400 P2 R | Port 2 | 16352* X1400 P3 | Port 3 | 16351* X1400 P4 | Port 4 | 16350* X127 | PN/E | 16374* X127 P1 | Port 1 | 16373* X142 | E/A | 66...105 | 66...105 **Hinweis** Es können nur Diagnoseadressen der Schnittstellen (**nicht der Ports!**) an den Funktionsbaustein übergeben werden. Werden falsche Diagnoseadressen an den Baustein übergeben, so können vom diesem keine Diagnosedaten ermittelt werden.
3.	Wechseln Sie in den SIMOTION SCOUT und markieren Sie im Ordner PROGRAMME der SIMOTION Steuerung die ST-Quelle `pMachineNetworkDiagnostics`. Im Symbolbrowser wird der Inhalt der ST-Quelle angezeigt.
4.	Öffnen Sie im Symbolbrowser die globale Struktur `gsNetworkAddressDetectionSettings` sowie das darin enthaltene Array `au32NetworkDiagnosticAddress`.  Name | Data type | Display format | Initial value | Status value | Control value 1 pMachineNetworkDiagnosticsPeripheralFaultTask | 'pMachine... 2 pMachineNetworkDiagnosticsBackgroundTask | 'pMachine... 3 gasMachineNetwork | 'ARRAY [1 4 gsNetworkAddressDetectionSettings | 'sAddress... 5 boAutoDetectionOfNetworkAddresses | BOOL | FALSE | TRUE 6 boManualNetworkAddressPresetting | BOOL | FALSE | FALSE 7 au32NetworkDiagnosticAddress | 'ARRAY [1... 8 au32NetworkDiagnosticAddress[1] | UDINT | DEC | 0 | 0 9 au32NetworkDiagnosticAddress[2] | UDINT | DEC | 0 | 0 10 au32NetworkDiagnosticAddress[3] | UDINT | DEC | 0 | 0 11 au32NetworkDiagnosticAddress[4] | UDINT | DEC | 0 | 0 12 au32NetworkDiagnosticAddress[5] | UDINT | DEC | 0 | 0 13 gboActualisationOfNetworkDiagnosticsStructure | BOOL | TRUE | FALSE 14 gasEventCounter | 'ARRAY [1... 15 gsCounterArray | 'sCounter... **Hinweis** Die Größe des Arrays ist abhängig vom Wert der Konstaten `NUMBER_OF_NETWORKS` in der ST-Quelle `fMachineNetworkDiagnostics`. Es können hier demnach nur maximal diese Anzahl an Netzwerk-Diagnoseadressen am Baustein vorgegeben werden.

Nr.	Aktion
5.	Steuern Sie mittels der Schaltfläche **„Steuern“** die Variable `boManualNetworkAddressPresetting` auf den Wert **„TRUE“**.  D435.pMachineNetworkDiagnostics: Symbol browser Control **Hinweis** Die Variable `boAutoDetectionOfNetworkAddresses` wird automatisch auf den Wert **„FALSE“** gesetzt.
6.	Tragen Sie anschließend die gewünschten Netzwerk-Diagnoseadressen in die Arrayelemente des Arrays `au32NetworkDiagnosticAddress` ein und übernehmen Sie diese ebenfalls mittels der Schaltfläche **„Steuern“**. **Hinweis** Es müssen nicht zwingend alle Arrayelemente mit einer Diagnoseadresse belegt werden. Für Elemente mit dem Wert **„0“** werden vom Baustein keine Diagnosedaten ermittelt, die entsprechenden Elemente der Anwenderstruktur enthalten somit ihre Standardwerte. Aus Performancegründen ist es allerdings sinnvoller in solch einem Fall den Wert der Konstanten `NUMBER_OF_NETWORKS` entsprechend anzupassen.
7.	Steuern Sie danach die globale Variable `gboActualisationOfNetworkDiagnosticsStructure` auf den Wert **„TRUE“**, um die Diagnosefunktionalität für die gewählten Netzwerke anzustoßen.  D435.pMachineNetworkDiagnostics: Symbol browser Control

Nr.	Aktion
8.	Es werden alle projektierten Stationen in den gewählten Netzwerken sowie deren aktueller Zustand ermittelt und in der Anwenderstruktur `gasMachineNetwork` hinterlegt. Die Reihenfolge der an den Baustein übergebenen Diagnoseadressen entspricht dabei der Reihenfolge der Daten, wie diese in der Anwenderstruktur hinterlegt werden. Wird z.B. die Netzwerk-Diagnoseadresse **16349** an das Arrayelement `au32NetworkDiagnosticAddress[2]` übergeben, so stehen nach Abarbeitung des Funktionsbausteins `FBInitNetworkDiagnosticStructure` die ermittelten Diagnosedaten für dieses Netzwerk auch im Arrayelement `gasMachineNetwork[2]` der Anwenderstruktur.

6 Literaturhinweise

6.1 Internet-Link-Angaben

Diese Liste ist keinesfalls vollständig und spiegelt nur eine Auswahl an geeigneten Informationen wieder.

Tabelle 6-1

	Themengebiet	Titel
\1\	Referenz auf den Beitrag	http://support.automation.siemens.com/WW/view/de/53705461
\2\	Siemens Industry Online Support	http://support.automation.siemens.com

7 Ansprechpartner

Siemens AG

Industry Sector
I DT MC PMA APC
Frauenauracher Straße 80
D - 91056 Erlangen
mailto: profinet.team.motioncontrol.i-dt@siemens.com

8 Historie

Tabelle 8-1

Version	Datum	Änderung
V1.3	12/2011	Erste Ausgabe
V2.0	06/2013	Überarbeitete Version, ST-Quellen und Dokumentation überarbeitet