SIEMENS

SIMOTION Meldungshandling

Applikationshandbuch

11/2016

Beitrags-ID: 48955585

Rechtliche Hinweise

Warnhinweiskonzept

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise zu Ihrer persönlichen Sicherheit sind durch ein Warndreieck hervorgehoben, Hinweise zu alleinigen Sachschäden stehen ohne Warndreieck. Je nach Gefährdungsstufe werden die Warnhinweise in abnehmender Reihenfolge wie folgt dargestellt.

⚠GEFAHR
bedeutet, dass Tod oder schwere Körperverletzung eintreten **wird**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠WARNUNG
bedeutet, dass Tod oder schwere Körperverletzung eintreten **kann**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

⚠VORSICHT
bedeutet, dass eine leichte Körperverletzung eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

ACHTUNG
bedeutet, dass Sachschaden eintreten kann, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Beim Auftreten mehrerer Gefährdungsstufen wird immer der Warnhinweis zur jeweils höchsten Stufe verwendet. Wenn in einem Warnhinweis mit dem Warndreieck vor Personenschäden gewarnt wird, dann kann im selben Warnhinweis zusätzlich eine Warnung vor Sachschäden angefügt sein.

Qualifiziertes Personal

Das zu dieser Dokumentation zugehörige Produkt/System darf nur von für die jeweilige Aufgabenstellung **qualifiziertem Personal** gehandhabt werden unter Beachtung der für die jeweilige Aufgabenstellung zugehörigen Dokumentation, insbesondere der darin enthaltenen Sicherheits- und Warnhinweise. Qualifiziertes Personal ist auf Grund seiner Ausbildung und Erfahrung befähigt, im Umgang mit diesen Produkten/Systemen Risiken zu erkennen und mögliche Gefährdungen zu vermeiden.

Bestimmungsgemäßer Gebrauch von Siemens-Produkten

Beachten Sie Folgendes:

⚠WARNUNG
Siemens-Produkte dürfen nur für die im Katalog und in der zugehörigen technischen Dokumentation vorgesehenen Einsatzfälle verwendet werden. Falls Fremdprodukte und -komponenten zum Einsatz kommen, müssen diese von Siemens empfohlen bzw. zugelassen sein. Der einwandfreie und sichere Betrieb der Produkte setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung, Montage, Installation, Inbetriebnahme, Bedienung und Instandhaltung voraus. Die zulässigen Umgebungsbedingungen müssen eingehalten werden. Hinweise in den zugehörigen Dokumentationen müssen beachtet werden.

Marken

Alle mit dem Schutzrechtsvermerk ® gekennzeichneten Bezeichnungen sind eingetragene Marken der Siemens AG. Die übrigen Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen kann.

Haftungsausschluss

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so dass wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
DEUTSCHLAND

Beitrags-ID: 48955585
℗ 11/2016 Änderungen vorbehalten

Inhaltsverzeichnis

Vorwort

1

1.1 Allgemeine Hinweise

Hinweis

Die Standardapplikationen sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit hinsichtlich Konfiguration und Ausstattung sowie jeglicher Eventualitäten. Die Standardapplikationen 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 Standardapplikationen entheben Sie nicht der Verpflichtung zu sicherem Umgang bei Anwendung, Installation, Betrieb und Wartung. Durch Nutzung dieser Standardapplikationen 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 Standardapplikationen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in diesen Standardapplikationen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Gewährleistung, Haftung und Support

Im Falle kostenloser Überlassung der Applikation gilt folgendes:

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

Andere und alle sonstigen Rechte und Ansprüche gegen die Siemens AG, gleich aus welchem Rechtsgrund, sind ausgeschlossen. Insbesondere Ansprüche gegen die Siemens AG auf Schadenersatz, insbesondere wegen Produktionsausfalls, Nutzungsausfalls, entgangenen Gewinns, direkter, indirekter oder Folgeschäden sind ausgeschlossen.

Dies gilt nicht, soweit zwingend gehaftet wird, z. B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, grober Fahrlässigkeit von Vorgesetzen und leitenden Angestellten der Siemens AG oder in Fällen des arglistigen Verschweigens von Mängeln.

Diese Haftungsbegrenzung findet auch Anwendung zu Gunsten von Subunternehmern, Zulieferern, Beauftragten, Vorgesetzten, leitenden Angestellten und Angestellten der Siemens AG.

Auf diese Vereinbarung findet für Kunden mit Firmensitz in Deutschland das deutsche Recht Anwendung für Kunden mit Firmensitz außerhalb Deutschlands findet das schweizerische Recht Anwendung. Die Anwendbarkeit des Übereinkommens der Vereinten Nationen über Verträge über den internationalen Warenkauf vom 11.04.1980 (CISG) ist ausgeschlossen.

Im Falle entgeltlicher Überlassung der Applikation gilt die für das jeweilige Geschäft zutreffende Alternative:

- Alternative 1: (Internes Geschäft)

Es gelten, sofern nicht unten etwas Abweichendes geregelt ist, die "Bedingungen für Lieferungen und Leistungen im Siemens-internen Geschäft" in der jeweils zum Zeitpunkt der Überlassung gültigen Fassung.

- Alternative 2: (Inlandsgeschäft der Siemens AG)

Es gelten, sofern nicht unten etwas Abweichendes geregelt ist, die "Allgemeine Bedingungen zur Überlassung von Software für Automatisierungs- und Antriebstechnik an Lizenznehmer mit Sitz in Deutschland" in der jeweils zum Zeitpunkt der Überlassung gültigen Fassung.

- Alternative 3: (Direktexportgeschäft der Siemens AG)

Es gelten, sofern nicht unten etwas Abweichendes geregelt ist, die "Allgemeine Bedingungen zur Überlassung von Softwareprodukten für Automation and Drives an Lizenznehmer mit Sitz außerhalb Deutschlands" in der jeweils zum Zeitpunkt der Überlassung gültigen Fassung.

Die Weitergabe oder Vervielfältigung dieser Applikationsbeispiele oder Auszüge daraus in unbearbeiteter Form sind nicht gestattet, soweit nicht ausdrücklich von Siemens Industry Sector gestattet.

Hinweis auf Exportkennzeichen

AL: N

ECCN: N

1.2 Über dieses Dokument

Ziel

Dieses Dokument soll bei der schnellen Integration der Applikation Meldungshandling für die Verwaltung von Meldungen in das vorhandene SIMOTION SCOUT Projekt helfen. Die Bibliothek mit dem Namen **LMsgHdl** stellt grundlegende Funktionalitäten zum Anzeigen und Verwalten von Meldungen bereit. Kenntnisse beim Umgang mit dem Engineeringsystem SIMOTION SCOUT werden vorausgesetzt.

Hinweis

Diese Dokumentation erhebt nicht den Anspruch, alle Gerätedetails oder Gerätevarianten zu erfassen oder jeden denkbaren Fall des Betriebes oder der Anwendung zu berücksichtigen.

Sollten Sie weitere Informationen benötigen oder sollten spezielle Probleme auftreten, die für das Anwendungsgebiet nicht ausführlich genug behandelt werden, wenden Sie sich bitte an die örtliche Siemens-Niederlassung.

Zielgruppe

Das vorliegende Dokument wendet sich an Programmierer, Inbetriebnehmer und Applikationsingenieure, die Anwendungen für SIMOTION erstellen.

Abgrenzung

Beim Meldungshandling werden Informationen aus SINAMICS Antrieben ausgelesen. Eine Beschreibung dieser Kommunikation ist nicht Bestandteil dieser Dokumentation.

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/48955585
(http://support.automation.siemens.com/WW/view/de/48955585)

Applikationsbeschreibung 2

2.1 Anwendungsgebiet

2.1.1 Beschreibung

In jeder SIMOTION Applikation treten Hinweise, Fehler, Alarme, Warnungen, Meldungen sowie anwenderdefinierte Meldungen auf. In diesem Dokument wird der Begriff **Meldungen** übergreifend verwendet. Meldungen können aufgrund von äußeren Einflüssen entstehen, wie z. B. Änderung des Status oder Fehlern von Peripheriegeräten oder des Antriebes. Meldungen können auch im Motion Control System SIMOTION ausgelöst werden. Solche sind beispielsweise Systemfehler oder Technologieobjekt-Fehler (technologische Alarme).

Alle Meldungen werden in einem Puffer (Log) gesammelt. Der Anwender hat die Möglichkeit, sich die aktuell anstehenden Meldungen sowie eine Meldunghistorie anzeigen zu lassen. Diese Informationen können an eine übergeordnete Steuerung oder ein Leitsystem weitergegeben werden. Die Meldungshistorie kann für eine Ferndiagnose, z. B. bei Störungen der Maschine, als Datei an einen System-Spezialisten zur Auswertung gesendet werden.

2.1.2 Einsatzbereich

Die Applikation Meldungshandling kann universell für beliebige Anwendungen eingesetzt werden. Es wird mindestens das Technologiepaket TP CAM benötigt.

2.2 Zielsetzung

2.2.1 Aufgabenstellung

Ziel des Meldungshandlings ist es, Meldungen aus verschiedenen Quellen des SIMOTION Systems zu sammeln und diese dem Anwender bereitzustellen. Unterschieden wird hierbei zwischen aktuell anstehenden Meldungen und einer Meldungshistorie.

Der Programmierer einer SIMOTION Applikation kann die gesammelten Meldungen weiterverarbeiten, indem diese entweder an ein überlagertes System weitergereicht oder auf einem HMI angezeigt werden.

Über ein Konfigurations-Skript soll das Meldungshandling in die SIMOTION Applikation integriert werden, indem Programme und Konstanten an die vorliegende Anwendung angepasst werden. Das Meldungshandling ist modular aufgebaut. Der Anwender soll bei

Aufruf des Konfigurations-Skriptes entscheiden können, welche Bestandteile eingefügt werden.

2.2.2 Nutzen

Mit dem vorliegenden Meldungshandling wird die Erstellungszeit eines Fehler- und Meldungshandlings, welches jede SIMOTION Applikation enthalten sollte, deutlich verkürzt.

Durch die Verwendung eines Konfigurations-Skripts für die Integration in die SIMOTION Applikation wird ein fehlerträchtiges Einfügen der Softwarebestandteile vermieden. Das Konfigurations-Skript liest die Projektinformationen ein und konfiguriert das Meldungshandling entsprechend.

Mit dem Meldungshandling werden Meldungen aus den folgenden Quellen gesammelt:

- Meldungen des SIMOTION Systems
  - Technologische Meldungen von SIMOTION
  - Meldungen über SIMOTION Systemfehler
  - Peripheriemeldungen
- Meldungen der SINAMICS Antriebe
  - DO-Safety-Meldungen
  - DO-Meldungen
- Anwenderdefinierte Meldungen

Die Meldungen werden in Puffern gespeichert, auf die der Anwender mit seinen Anwendungen zugreifen kann. Diese sind:

- Puffer für speicheroptimierte Daten, die für eine weitere Verwendung vom Anwender aufbereitet werden müssen
- stringbasierte Puffer, die gut lesbar, dafür aber speicherintensiv sind

Zusätzlich können die Anzeigeverfahren AlarmS und Bitmeldeverfahren verwendet werden.

Neben der Sammlung von Meldungen kann der Anwender für anwenderdefinierte Meldungen und Peripheriemeldungen parametrieren, wie schwerwiegend sich eine Meldung auf die Funktionstüchtigkeit der Maschine auswirkt. In Abhängigkeit der Schwere der Meldung, kann vom Anwender eine Maschinenreaktion programmiert werden.

2.3 Konzept

2.3.1 Darstellung des Konzeptes

Meldungen sammeln und anzeigen

Für das Meldungshandling wurde der Ansatz verfolgt, alle Meldungen aus den unterschiedlichen Quellen in ein einheitliches Format zusammenzufassen. Dadurch können alle Meldungen des SIMOTION Systems auf einem HMI angezeigt oder an eine übergeordnete Steuerung gesendet werden. Weiterhin kann der Anwender bei einer weiteren Verarbeitung auf eine einheitliche Schnittstelle zugreifen.

In der folgenden Abbildung ist dargestellt, wie die Meldungen aus den verschiedenen Quellen gesammelt werden. Alle Meldungen werden durch einen Meldungssammler eingelesen und in ein einheitliches Format umgewandelt.

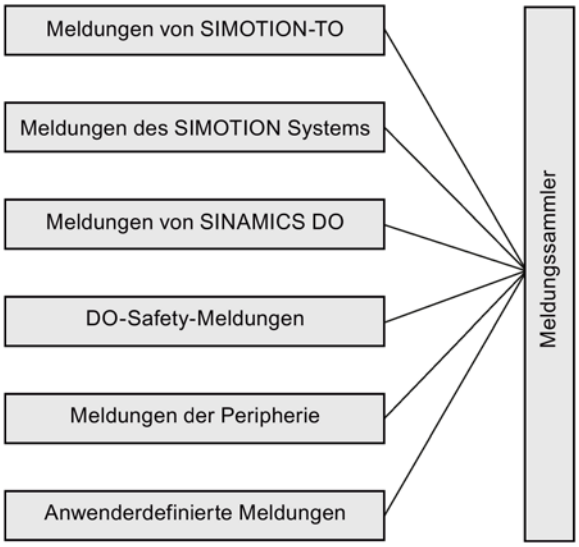


Bild 2-1 Meldungen sammeln

Hinweis

Meldungen des SIMOTION Systems umfassen Meldungen, die durch Zeitüberläufe in den TimeFaultTasks sowie durch Fehler beim Verarbeiten eines Programms in der ExecutionFaultTask generiert werden.

Der Meldungssammler schreibt die Meldungen in zwei Puffer

- einen für die aktuell anstehenden Fehler und
- einen Puffer mit der Meldungshistorie

Werden die Meldungen durch das Meldungshandling quittiert, wird der Puffer mit den aktiven Meldungen geleert. Die Meldungen in der Meldungshistorie werden als *Meldung gegangen*

markiert (Zeitstempel, wann quittiert wurde). Ist der Puffer der Meldungshistorie voll, wird die älteste Meldung überschrieben.

Das Speicherformat dieser beiden Puffer wird im Weiteren als **Rohdaten** bezeichnet, da diese beiden Puffer auf möglichst hohe Speichereffizienz optimiert sind. Die Meldungsinformationen sind bei den Rohdatenpuffern mit Zahlenwerten kodiert. Der Puffer für die Meldungshistorie wird im netzausfallsicheren Datenbereich (RETAIN) gespeichert, um auch nach einem Spannungsausfall verfügbar zu sein.

Damit der codierte Zahlenwert der Rohdaten besser lesbar wird, können zwei weitere Puffer integriert werden. Diese enthalten die Meldungsinformationen im Format String und können optional durch das Meldungshandling erstellt werden. Entsprechend zu den Rohdatenpuffern gibt es einen Stringpuffer für aktive Meldungen und einen Puffer für die Meldungshistorie. Diese Puffer im Format String benötigen wesentlich mehr Speicher als die Rohdatenpuffer.

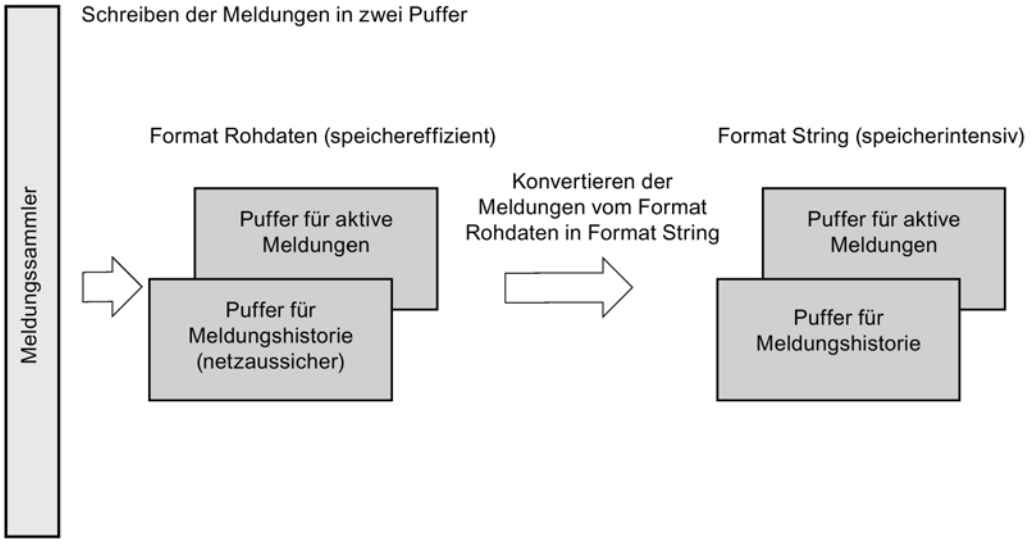


Bild 2-2 Meldungen in Puffer schreiben

Für die Anzeige von anwenderdefinierten Meldungen auf dem HMI können das AlarmS- und das Bitmeldeverfahren zusätzlich verwendet werden.

Der Anwender kann auf die Puffer zugreifen. Die Puffer können in beliebigen Teilausschnitten über ebenfalls bereitgestellte Funktionsbausteine direkt auf dem HMI angezeigt werden. Bei der Verwendung der Rohdatenpuffer sollte das HMI aus den Rohdaten verständliche Fehlertexte generieren. Die Texte aus den Stringpuffern können direkt für die Anzeige verwendet werden. Weiterhin können die Puffer an eine übergeordnete Steuerung beispielsweise über TCP/IP übertragen werden.

Eine Ferndiagnose kann erleichtert werden, indem der Anwender den Puffer mit der Meldungshistorie bei einem Maschinenausfall auf das Speichermedium des SIMOTION Gerätes speichert und diese Daten an einen System-Spezialisten übermittelt.

Hinweis

HMI-Masken zur Anzeige der Meldungen im Format STRING und SIMOTION Programme für das Senden an eine übergeordnete Steuerung sind nicht Bestandteil des Meldungshandlings. Dies ist Aufgabe des Anwenders.

Da das Meldungshandling modular aufgebaut ist, kann der Anwender auswählen, ob das komplette Meldungshandling oder nur Teile davon in die Applikation übernommen werden sollen. So ist es dem Anwender möglich auszuwählen, aus welchen Quellen Meldungen gesammelt, welche Puffer verwendet und auf welche Art Meldungen angezeigt werden.

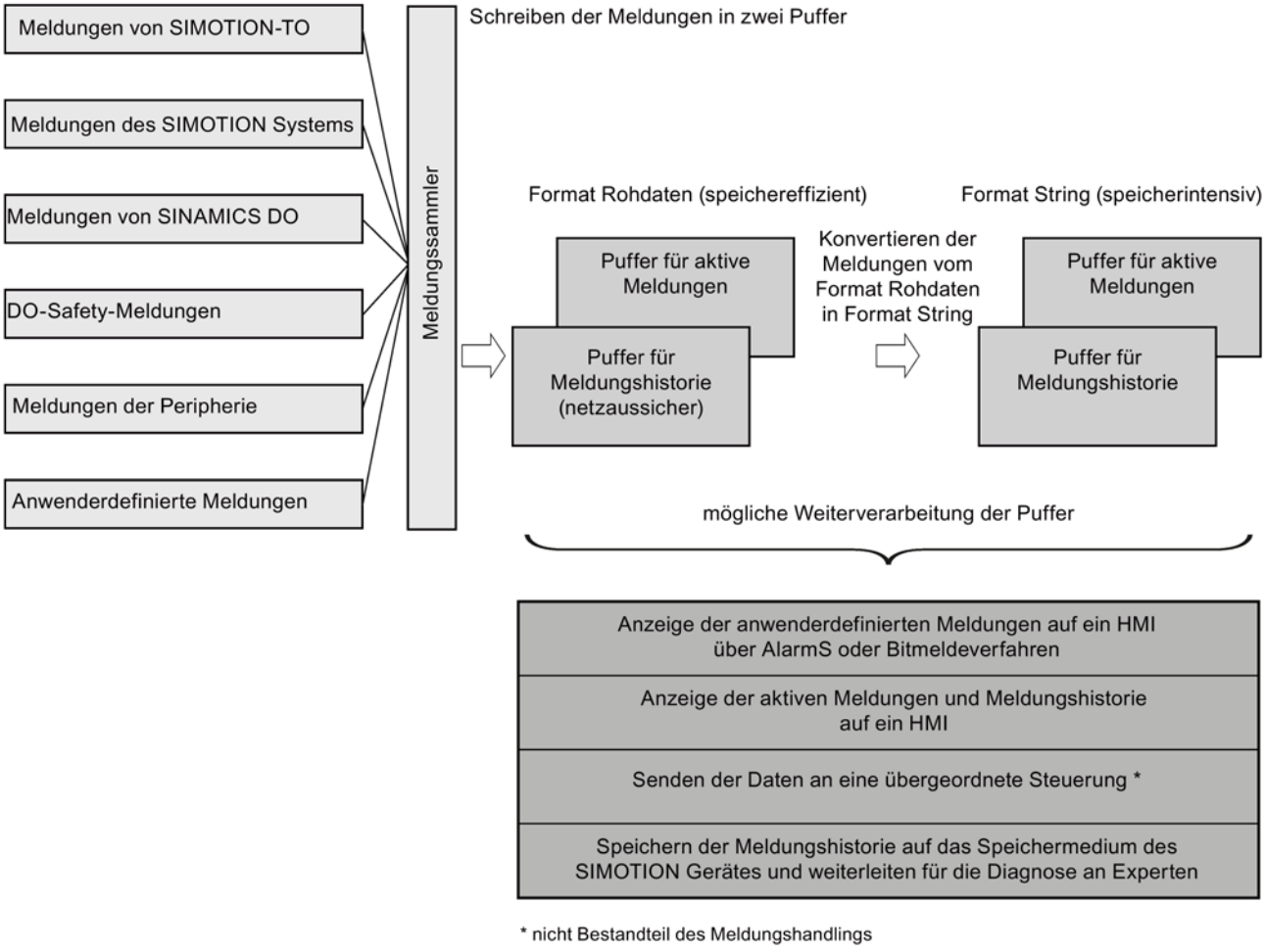


Bild 2-3 Gesamtübersicht Meldungshandling

Reaktion auf Meldungen

Bisher wurde beschrieben, wie Meldungen gesammelt und zur Anzeige gebracht werden. Die meisten Meldungen erfordern aber auch von der Maschine eine Reaktion. Diese Reaktion ist von der Art und der Quelle der Meldung abhängig. Der Ausfall eines wichtigen Equipment Modules (Maschinen-Aggregats) wie z. B. beim Auftreten eines Schleppabstandsfehlers an der Achse eines synchron laufenden Equipment Modules erfordert eine andere Reaktion als der Ausfall eines Equipment Modules zum Befüllen eines Materialspeichers. Im ersten Fall wäre ein Nothalt der Maschine notwendig, im zweiten Fall könnte noch so lange produziert werden, wie Material vorrätig ist.

Hinweis

Reaktionen auf technologische Alarme und Antriebsfehler werden im Engineeringsystem SIMOTION SCOUT eingestellt und sind nicht Bestandteil des Meldungshandlings.

Jeder Meldung aus den anwenderdefinierten Meldungen und den Peripheriemeldungen kann eine Maschinenfehlerklasse zugewiesen werden. Diese Maschinenfehlerklasse bestimmt das Verhalten der Maschine nach Auftreten von Meldungen. Bei mehreren gleichzeitigen Meldungen bestimmt die Fehlerklasse mit der höchsten Priorität die Maschinenfehlerklasse. Detaillierte Beschreibung siehe Abschnitt Maschinenfehlerklassen definieren (Seite 66).

Hinweis

Die Reaktion bei den Maschinenfehlerklassen ist vom Anwender zu programmieren. Das Meldungshandling liefert hierfür eine Variable für die Maschinenfehlerklasse der höchsten Priorität. In einer Variablen sind alle derzeit anstehenden Maschinenfehlerklassen zu entnehmen.

2.4 Systemübersicht (Beispiel)

2.4.1 Automatisierungsübersicht (Beispiel)

Das Meldungshandling sammelt Meldungen von der am SIMOTION Gerät angeschlossenen Peripherie ein. Im folgenden Bild ist beispielhaft eine Automatisierungslösung mit einem SIMOTION D Gerät dargestellt. An das Meldungshandling werden Meldungen der am PROFIBUS oder PROFINET angeschlossenen Peripheriegeräte (im Bild eine ET200M, eine ET200S sowie ein SINAMICS S120 CU320) weitergereicht.

Weiterhin kann bei SINAMICS Antrieben auf die Meldungen von Antriebsobjekten (DOs) zugegriffen werden. Hierbei kann der SINAMICS Antrieb sowohl über den integrierten PROFIBUS (bei SIMOTION D ist das der SINAMICS Integrated) als auch über PROFIBUS oder PROFINET angeschlossen sein. Die Kommunikation erfolgt über azyklische Dienste. Dadurch ist eine detaillierte Suche nach Fehlern bei SINAMICS Antrieben möglich.

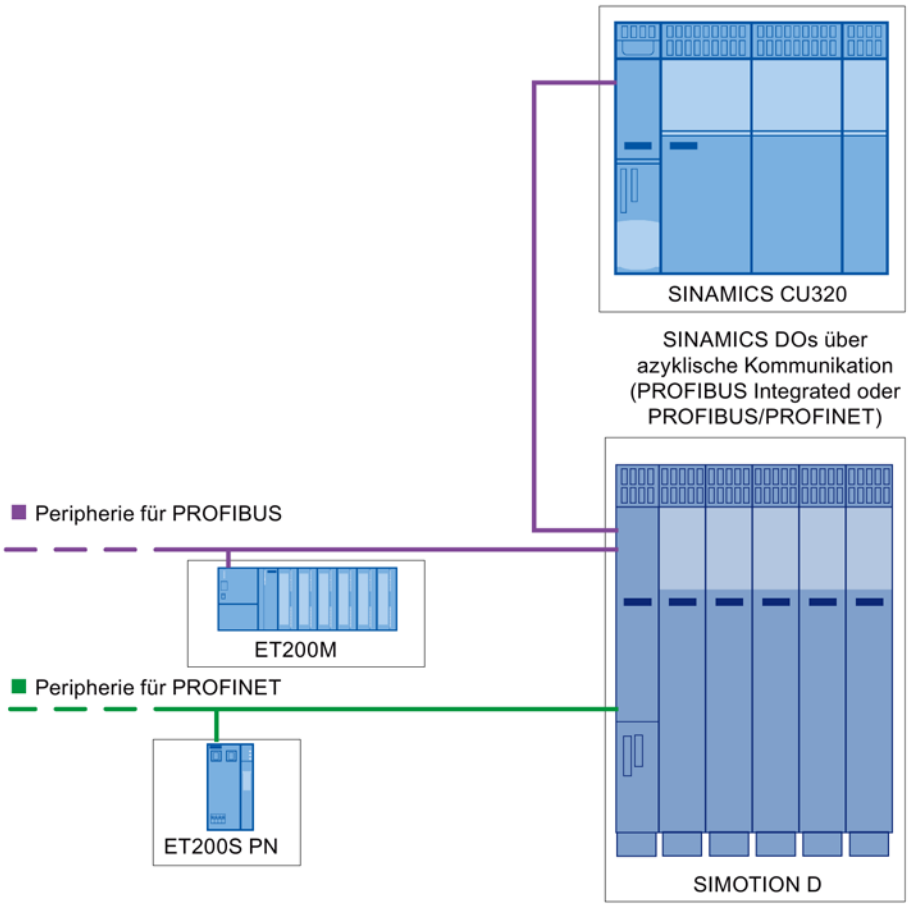


Bild 2-4 Beispiel für Automatisierungslösung

2.4.2 Hardwarestruktur

Das Meldungshandling wurde für das Motion Control System SIMOTION erstellt und setzt eine Steuerung dieses Typs voraus. Sie kann für alle Ausprägungen von SIMOTION Geräten (SIMOTION D, C, und P) verwendet werden. Der Antriebstyp ist unter Verwendung eines PROFIdrive Standardtelegrammes zu einem Antrieb SINAMICS S120 prinzipiell unerheblich, allerdings wird direkt auf das Zustandswort des Antriebes zurückgegriffen, was bei einer anders definierten Schnittstelle unter Umständen nicht funktioniert.

2.4.3 Systemvoraussetzungen

Das Meldungshandling wurde für die Softwareversion ab SIMOTION SCOUT V4.1 SP4 mit SINAMICS 2.5 und 2.6 erstellt und getestet. Es wird mindestens das Technologiepaket TP CAM benötigt.

2.4.4 Lieferumfang

Auf dem Auslieferungsmedium finden Sie folgende Daten:

- die Bibliotheken LDPV1 und LMsgHdl im XML-Format
- die Programmunits des Meldungshandlings im XML-Format
- SIMOTION IT Seiten
- Dateien im XML-Format zur Sprachumschaltung der Meldungstexte in Deutsch, Englisch, Französisch und Italienisch

Applikationsstruktur

3

3.1 Struktur der Bibliotheken

3.1.1 Übersicht der Bibliotheken

Für das Meldungshandling werden folgende Bibliotheken verwendet:

- Bibliothek **LDPV1** für azyklische Kommunikation mit SINAMICS Antrieben.
  Die Beschreibungen zu Bausteinen und Funktionalitäten dieser Bibliothek befinden sich auf dem Speichermedium Utilities & Applications, die Bestandteil von SIMOTION SCOUT ist.
- Bibliothek **LMsgHdl** für Funktionalitäten des Meldungshandlings. Die Bibliothek ist in verschiedene Units gegliedert. Die Bibliothek LMsgHdl ist für das Meldungshandling erstellt und wird in diesem Dokument beschrieben.

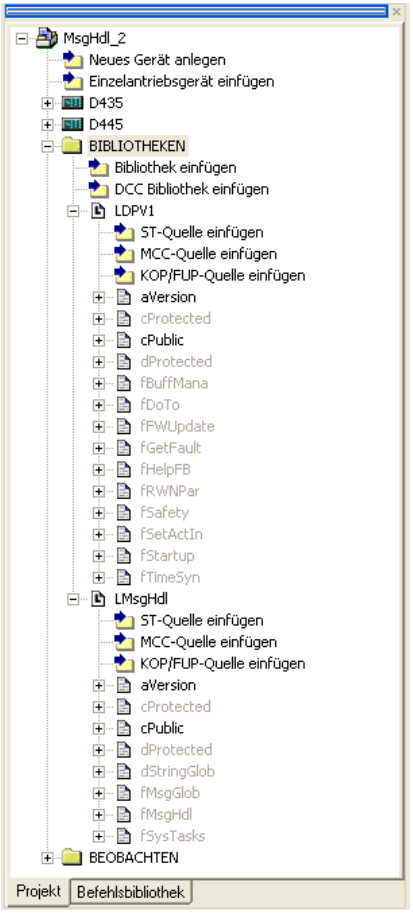


Bild 3-1 Übersicht der Bibliotheken für das Meldungshandling

3.1.2 Struktur der Bibliothek LMsgHdl

In der folgenden Tabelle sind die Units der Bibliothek **LMsgHdl** aufgeführt. Der Anwender hat Zugriff auf zwei Units der Bibliothek, alle anderen sind mit einem Know-how-Schutz versehen. Es werden nur die Units ohne Know-how-Schutz beschrieben.

Die Unit **aVersion** dient zur Kennzeichnung der Versionshistorie der Bibliothek. Es ist kein Quellcode in dieser Unit enthalten.

Die Konstanten der Unit **cPublic** werden im Abschnitt Konstanten (Seite 22) beschrieben.

Tabelle 3- 1 Aufbau der Bibliothek LMsgHdl

Unitname	Verwendung	Know-how-Schutz
aVersion	Unit der Versionsübersicht, Änderungsliste	nein
cProtected	Unit der Definition der geschützten Konstanten	ja
cPublic	Unit der Definition der vom Anwender änderbaren Konstanten	nein
dProtected	Unit der geschützten Daten	ja
dStringGlob	Unit für Texte der stringbasierten Puffer auf Deutsch, Englisch, Französisch und Italienisch	ja
fMsgGlob	Unit für Funktionen für stringbasierte Puffer	ja
fMsgHdl	Unit für Funktionen des Meldungshandlings	ja
fSysTasks	Unit für Funktionen zum Auslesen von Meldungen aus SIMOTION Fehlertasks	ja

3.2 Struktur der Units im SIMOTION Projekt

Units des Meldungshandlings

In der SIMOTION Applikation werden durch das Ausführen des Konfigurations-Skripts des Meldungshandlings Units angelegt. Die Schnittstellen zur Bedienung des Meldungshandlings sowie die Meldungspuffer sind in globalen Variablen definiert. Folgende Units sind in der Applikation des Meldungshandlings vorhanden:

Tabelle 3- 2 Units des Meldungshandlings in der SIMOTION Applikation

Unit	Verwendung	Know-how-Schutz
fLMsgHdlInit	Funktionen die durch das Konfigurations-Skript, bzw. vom Anwender an das jeweilige Projekt angepasst werden müssen.	nein
fLMsgHdl	Funktionen für das Meldungshandling	ja
pLMsgHdl	Unit für Programme des Meldungshandlings	ja
dLMsgHdl	alternative Meldungsvariante (standardmäßig nicht aktiviert)	nein

Hinweis

Die in den Programmunits deklarierten globalen Variablen können in der Anwenderapplikation verwendet und über den Symbolbrowser beobachtet werden. Eine Beschreibung der globalen Variablen finden Sie im Übersicht der globalen Variablen (Seite 111).

Programme im Meldungshandling

In der Unit **pLMsgHdl** sind Programme enthalten, die vom Konfigurations-Skript in das Ablaufsystem eingehängt werden. Diese dienen zur Initialisierung, zum Meldungen sammeln und um Meldungen in den Puffern zu verarbeiten. Die Unit ist mit einem Know-how-Schutz versehen, es können keine Änderungen vom Anwender vorgenommen werden.

Tabelle 3- 3 Programme in der Unit pLMsgHdl

Name des Programmes	Taskebene	Verwendung
pLMsgHdlStartupMessageHandling	StartupTask	• Initialisierung der Daten • Zuweisung der TO-Referenzen, der DO-Adressen und der Peripherieadressen • Einstellung der Maschinenfehlerklassen
pLMsgHdlTechnologicalMessage	TechnologicalFaultTask	Einlesen der technologischen Meldungen
pLMsgHdlPeripheralMessage	PeripheralFaultTask	Einlesen der Peripheriemeldungen
pLMsgHdlTimeFaultMessage	TimeFaultTask	Einlesen von Meldungen bei Zeitüberlauf
pLMsgHdlTimeFaultBackgroundMessage	TimeFaultBackgroundTask	Einlesen von Meldungen bei Zeitüberlauf in der BackgroundTask
pLMsgHdlExecutionFaultMessage	ExecutionFaultTask	Einlesen von Meldungen bei Durchlauf der SystemFaultTask
pLMsgHdlMain	BackgroundTask	Aufruf aller Programme des Meldungshandlings in der BackgroundTask

3.3 Konstanten

3.3.1 Öffentliche Konstanten

Die folgenden Konstanten werden durch das Konfigurations-Skript belegt und sollten vom Anwender nicht verändert werden.

Hinweis

Die Konstanten in der Unit **cPublic** für die Anzahl der TOs und der DOs im Projekt werden vom Konfigurations-Skript auf ihre korrekten Werte geschrieben. Diese Konstanten dürfen nicht vom Anwender verändert werden. Ändert sich die Konfiguration des Projektes, muss das Konfigurations-Skript erneut aufgerufen werden. Anschließend wird bei einem Neustart des SIMOTION Gerätes der MeldungsLog im netzausfesten Datenbereich (RETAIN) neu initialisiert.

Tabelle 3- 4 Öffentliche Konstanten in der Unit **cPublic** der Bibliothek **LMsgHdl** (vom Konfigurations-Skript belegt)

Name	Wert	Verwendung
LMSGHDL_LENGTH_OF_MESSAGE_LOG	200	Länge des Puffers für die Meldungshistorie (gilt für Rohdaten- und Stringpuffer)
LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES	100	Länge des Puffers für aktive Meldungen (gilt für Rohdaten- und Stringpuffer)
LMSGHDL_LANGUAGE_FOR_MESSAGE_STRING	9	Einstellung der Sprache (Defaulteinstellung 9 = englisch) (STEP 7 - Notation)
LMSGHDL_NUMBER_OF_AXES	1	Anzahl der Achsen im Projekt (reale und virtuelle)
LMSGHDL_NUMBER_OF_EXTERNAL_ENCODERS	1	Anzahl der externen Geber im Projekt
LMSGHDL_NUMBER_OF_MEASURING_INPUTS	1	Anzahl der Messtaster im Projekt
LMSGHDL_NUMBER_OF_OUTPUT_CAMS	1	Anzahl der Nocken im Projekt
LMSGHDL_NUMBER_OF_CAM_TRACKS	1	Anzahl der Nockenspuren im Projekt
LMSGHDL_NUMBER_OF_CAMS	1	Anzahl der Kurvenscheiben im Projekt
LMSGHDL_NUMBER_OF_FOLLOWING_OBJECTS	1	Anzahl der Folgeobjekte im Projekt
LMSGHDL_NUMBER_OF_PATH_OBJECTS	1	Anzahl der Bahnobjekte im Projekt
LMSGHDL_NUMBER_OF_FIXED_GEARS	1	Anzahl der Festen Getriebe im Projekt
LMSGHDL_NUMBER_OF_ADDITION_OBJECTS	1	Anzahl der Addierobjekte im Projekt
LMSGHDL_NUMBER_OF_FORMULA_OBJECTS	1	Anzahl der Formelobjekte im Projekt
LMSGHDL_NUMBER_OF_SENSORS	1	Anzahl der Sensorobjekte im Projekt
LMSGHDL_NUMBER_OF_CONTROLLER_OBJECTS	1	Anzahl der Reglerobjekte im Projekt
LMSGHDL_NUMBER_OF_TEMPERATURE_CONTROLLERS	1	Anzahl der Temperaturregler im Projekt
LMSGHDL_NUMBER_OF_TOS_WITH_DO	1	Anzahl der DOs mit Technologieobjekt (elektrische Achsen)

Name	Wert	Verwendung
LMSGHDL_NUMBER_OF_CYCLIC_DOS	1	Anzahl der DOs mit zyklischer Kommunikation (ohne DOs die mit dem TO Achse verbunden sind)
LMSGHDL_NUMBER_OF_ACYCLIC_DOS	1	Anzahl der DOs mit azyklischer Kommunikation (kein projektiertes Telegramm)
LMSGHDL_NUMBER_OF_PERIPHERAL_DEVICES	1	Anzahl der Peripheriegeräte
LMSGHDL_MAX_NUMBER_OF_SYSTEM_TASKS	53	Anzahl vorhandener Tasks im Ablaufsystem des SIMOTION Projektes

Vom Anwender verwendbare, nicht editierbare Konstanten

Die folgenden Konstanten können vom Anwender verwendet werden, um die anwenderdefinierten Meldungstexte zu editieren. Diese Konstanten dürfen **nicht** vom Anwender verändert werden.

Tabelle 3- 5 Öffentliche Konstanten in der Unit **cPublic** der Bibliothek **LMsgHdl** (vom Anwender verwendbar)

Name	Wert	Verwendung
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_1	1	Kann zur Übergabe von Zusatzwert 1 (additionalValue1) für eine anwenderdefinierte Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_2	2	Kann zur Übergabe von Zusatzwert 2 (additionalValue1) für eine anwenderdefinierte Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_FB_ID	3	Kann zur Übergabe von Zusatzwert FB-ID (functionBlockId) für eine anwenderdefinierte Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_ERROR_CODE	4	Kann zur Übergabe von Zusatzwert Fehlerquelle (errorCode) für eine anwenderdefinierte Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_REAL	5	Kann zur Übergabe von Zusatzwert REAL (additionalValueReal) für eine anwenderdefinierte Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT	0	Kann zur Übergabe von Datentyp DINT für die Ausgabe einer anwenderdefinierten Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_HEX	1	Kann zur Übergabe von Datentyp HEX für die Ausgabe einer anwenderdefinierten Meldung verwendet werden.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_REAL	2	Kann zur Übergabe von Datentyp REAL für die Ausgabe eines Beiwerts in einer anwenderdefinierten Meldung verwendet werden.

3.3.2 Veränderbare öffentliche Konstanten

Die folgenden Konstanten werden nicht durch das Konfigurations-Skript belegt und müssen je nach Anforderung vom Anwender verändert werden.

Hinweis

Der Wert der Konstanten muss mindestens 1 (Eins) betragen. Der Wert 0 (Null) ist nicht zulässig.

Tabelle 3- 6 Veränderbare öffentliche Konstanten in der Unit **cPublic** der Bibliothek **LMsgHdl**

Name	Wert	Verwendung
LMSGHDL_NUMBER_OF_STRING_MESSAGES_PER_CYCLE_IN_STARTUP		
	3	Anzahl der Meldungen im Format String, die im Hochlauf bei der Initialisierung gebildet werden. Über diese Konstante kann die Hochlaufzeit des Meldungshandlings verkürzt werden. Wird der Wert zu groß gewählt, kann es jedoch zum Zeitüberlauf in der Background Task kommen.
LMSGHDL_MAX_NUMBER_OF_NEW_MESSAGES_PER_CYCLE		
	1	Anzahl der neuen Meldungen, die in einem Backgroundzyklus neu in den Meldungspuffer übernommen werden können. Hier sollte nach Möglichkeit der Defaultwert beibehalten werden.
LMSGHDL_AUTO_SAVE_MESSAGE_BUFFER_TO_STORAGE_MEDIUM		
	FALSE	Aktivierung / Deaktivierung der Funktionalität automatisches Speichern des Meldungspuffers auf das Speichermedium des SIMOTION Geräts
LMSGHDL_MAX_NUMBER_OF_DATASETS_ON_STORAGE_MEDIUM		
	5	Anzahl der Dateien, die bei Auto-Save = TRUE angelegt werden
LMSGHDL_NUMBER_OF_EXECUTION_FAULT_MESSAGES		
	2	Größe des Zwischenpuffers zum Sammeln der Meldungen durch Programmausführungsfehler zur Laufzeit. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_TECH_FAULT_MESSAGES		
	100	Größe des Zwischenpuffers zum Sammeln der Technologiemeldungen. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_PERIPHERAL_FAULT_MESSAGES		
	50	Größe des Zwischenpuffers zum Sammeln der Peripheriemeldungen. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_TIME_FAULT_MESSAGES		
	5	Größe des Zwischenpuffers zum Sammeln der Meldungen durch Zeitüberlauf. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_APPLICATION_MESSAGES		
	20	Größe des Zwischenpuffers zum Sammeln der anwenderdefinierten Meldungen innerhalb einer Task. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_DO_FAULT_MESSAGES		
	50	Größe des Zwischenpuffers zum Sammeln der Fehler an Antriebsobjekten. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_DO_ALARM_MESSAGES		

Name	Wert	Verwendung
	50	Größe des Zwischenpuffers zum Sammeln der Warnungen an Driveobjekten. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_NUMBER_OF_DO_SAFETY_MESSAGES		
	50	Größe des Zwischenpuffers zum Sammeln der Safety-Meldungen an Antriebsobjekten. Bei Überlauf kann hier der Zwischenpuffer vergrößert werden.
LMSGHDL_ALARM_S_USER_MESSAGES		
	FALSE	TRUE: Verwendung von AlarmS zur Meldungsanzeige auf dem HMI FALSE: keine Verwendung von AlarmS
LMSGHDL_MESSAGE_BIT_USER_MESSAGES		
	FALSE	TRUE: Verwendung des Bitmeldeverfahrens zur Meldungsanzeige auf dem HMI FALSE: keine Verwendung des Bitmeldeverfahrens
LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI		
	10	Anzahl der maximalen Zeilen für die Anzeige eines Meldungspuffers (tatsächliche Anzahl wird für MeldungsLog und Anzeige der aktiven Fehler jeweils gesondert übergeben)
LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI		
	80	Maximale Zeichenlänge für Meldungstexte im Format String, die über die internen Funktionsbausteine an ein HMI bzw. SIMOTION IT übergeben werden können.
LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS		
	10	Anzahl der anwenderdefinierten Meldungen
LMSGHDL_NUMBER_OF_FUNCTION_BLOCK_IDS		
	1	Anzahl der anwenderdefinierten Meldungen durch FBs / FCs. Diese Meldungen von FBs / FCs zählen zu der Gesamtanzahl der Meldungen in der Konstanten LMSGHDL_MAX_NUMBER_OF_USER_DEFINED_EVENTS
LMSGHDL_MACHINE_ERROR_CLASS_ERROR_IN_MESSAGEHANDLING		
	0	Gibt die Meldungsklasse vor, die Meldungen die vom Meldungshandling selber abgesetzt werden, im Meldungshandling setzt.

3.4 Kernfunktionen und Bestandteile

3.4.1 Übersicht der Kernfunktionen und notwendigen Bestandteile des Meldungshandlings

Das Meldungshandling besitzt folgende Kernfunktionen:

- Meldungen verwalten und anzeigen
- Meldeverfahren AlarmS oder Bitmeldeverfahren nutzen
- aktive Meldungen quittieren
- aktuelle Meldungen auf das Speichermedium des SIMOTION Gerätes speichern
- Sprache von Meldungstexten einstellen

Sonstige notwendige Bestandteile im Meldungshandling:

- Pufferverwaltung für azyklische Kommunikationsdienste DP-V1
- Hochlaufprüfung des SIMOTION Gerätes

3.4.2 Beschreibung der Kernfunktionen und notwendigen Bestandteile

3.4.2.1 Pufferverwaltung

Pufferverwaltung für azyklische Kommunikationsdienste DPV1

Das Meldungshandling verwendet bei der Ermittlung von Informationen auf SINAMICS Baugruppen den azyklischen Kommunikationsdienst DPV1. Um eine Kollision der einzelnen Kommunikationsaufträge zu vermeiden, verwendet das Meldungshandling die globale Pufferverwaltung (Programm *pGlobalBufferManager*) der Bibliothek **LDPV1**. Diese Vorgehensweise ist erforderlich, da je SINAMICS Device nur ein azyklischer Kommunikationsauftrag gleichzeitig bearbeitet werden kann.

Beim Einsatz des Meldungshandlings ist demnach zwingend darauf zu achten, dass in der gesamten Applikation alle weiteren azyklischen Kommunikationsaufträge ebenfalls über die globale Pufferverwaltung der Bibliothek LDPV1 abgesetzt werden. Das Meldungshandling verwendet den Puffer mit der Kennung Null (0) für alle Antriebsgeräte, d. h. es dürfen in Applikationen, in denen azyklische Kommunikation verwendet wird, keine weiteren Puffer verwendet werden, um darüber Zugriffe zu koordinieren. Detaillierte Informationen finden Sie in der Dokumentation zur Bibliothek **LDPV1**.

Um Kollisionen in der Kommunikation zu vermeiden, sollten anstelle von Systembefehlen für die Kommunikation im Antrieb die Funktionen und Funktionsbausteine der Bibliothek **LDPV1** verwendet werden. Diese Bausteine sind bereits zur Verwendung der Pufferverwaltung ausgelegt.

Hinweis

Es muss sichergestellt sein, dass das Programm *pGlobalBufferManager* im Projekt vorhanden und der BackgroundTask zugeordnet ist. Das Programm wird zusammen mit der LDPV1 Bibliothek ausgeliefert.

3.4.2.2 Beschreibung der Puffer

Allgemein

Im Meldungshandling werden vier Puffer angelegt. Je nach Einstellung im Konfigurations-Skript, werden nur die Puffer für die Rohdaten oder alle vier Puffer mit Daten versorgt:

- Puffer für MeldungsLog als Rohdaten (Meldungshistorie)
- Puffer für aktive Meldungen Rohdaten
- Puffer für MeldungsLog als String (Meldungshistorie Strings), optional
- Puffer für aktive Meldungen String, optional

Diese Puffer werden in den folgenden Abschnitten beschrieben. Für die Interpretation der Meldungen im Format Rohdaten, siehe Abschnitt Interpretation der Rohdaten (Seite 121).

Hinweis

Mit der Funktion *AutoSave* wird das MeldungsLog gespeichert, sobald der Puffer voll ist. Alle Meldungen, die zu diesem Zeitpunkt keinen Zeitstempel *gegangen* haben, werden ohne diesen Zeitstempel gespeichert.

Puffer für MeldungsLog als Rohdaten (Meldungshistorie)

Im Puffer für das gesamte MeldungsLog werden alle aufgetretenen Meldungen in Form von Rohdaten angezeigt. Diese Daten werden als globale Daten im netzausfesten Bereich (RETAIN) abgelegt:

- Daten der Puffer für aktive Meldungen
- Zeitstempel **Meldung gegangen**

Ist der Speicherbedarf der Meldungshistorie größer als der Datenbereich RETAIN, kann der Datenbereich für die SIMOTION Gerät von RETAIN nach nicht RETAIN gewechselt werden. Dazu ist in der Programmunit fLMsgHdl die Preprozessor Definition LMSGHDL_NO_RETAIN_BUFFER zu setzen, siehe dazu Meldungen unterdrücken (Seite 59).

Da die meldungsspezifischen Informationen je nach Quelle stark voneinander abweichen, werden diese Informationen in einer allgemein definierten Struktur gesammelt. Dies hat zur Folge, dass nicht immer alle Elemente dieser Struktur gefüllt sind.

Um diese Meldungsinformationen auswerten zu können, muss auf der auswertenden Seite bekannt sein, wie die einzelnen Meldungen der unterschiedlichen Quellen zu interpretieren sind.

Da der Puffer für den MeldungsLog an ein HMI übertragen werden kann, wird für den Puffer der Aufbau in STRUCT OF ARRAY verwendet. Dadurch wird eine höhere Performance erzielt. Dieser Puffer befindet sich in der Programmunit **fLMsgHdl** und hat die Bezeichnung *grsLMsgHdlMessageLogBaseData*.

- Pro Meldung werden im Puffer für das gesamte MeldungsLog 64 Byte benötigt.
- Die Länge des Speichers für das gesamte MeldungsLog ist über die Konstante LMSGHDL_LENGTH_OF_MESSAGE_LOG einstellbar. Die Voreinstellung liegt bei 200 Einträgen.
  Somit benötigt das gesamte MeldungsLog ca. 12 kByte Speicher im netzausfesten Datenbereich (RETAIN).
  Ausnahme bei SIMOTION D410, da beträgt die Voreinstellung 150 Einträge.

Der Puffer für MeldungsLog als Rohdaten ist als Ringpuffer aufgebaut.

Puffer für aktive Meldungen Rohdaten

Im Puffer für aktive Meldungen werden alle noch nicht quittierten, bzw. nicht quittierbaren Meldungen angezeigt. Diese Daten werden als globale Daten abgelegt.

Für jede Meldung werden folgende Informationen abgelegt:

- Kennung für Quelle der Meldung
- Stufe der Meldung (Fehler, Störung, Warnung)
- Quittierungsart der Meldung, z. B. benötigen manche DO-Meldungen ein PowerOn als Quittierung
- Meldungsklasse
- Meldungsinformationen als Rohdaten, so wie sie in der Anwendung auftreten
- Zeitstempel **Meldung gekommen**

Der Aufbau und die Verwendung der Daten ist identisch mit denen der Meldungshistorie im Format Rohdaten. Es fehlt lediglich der Zeitstempel **Meldung gegangen**.

Dieser Puffer befindet sich in der Programmunit **pLMsgHdl** und hat die Bezeichnung *gsLMsgHdlActiveMessagesBaseData*.

- Pro Meldung werden im Puffer für die aktiven Meldungen 51 Byte benötigt.
- Die Länge des Speichers für die aktiven Meldungen ist über die Konstante LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES einstellbar. Die Voreinstellung liegt bei 100 Einträgen.
  Somit benötigt der gesamte Meldungspuffer für aktive Meldungen als Rohdaten ca. 5,1 kByte Speicher im globalen Datenbereich.

Der Puffer für aktive Meldungen ist kein Ringpuffer und wird nacheinander mit den gefundenen Einträgen im MeldungsLog beschrieben. Sind mehr Meldungen aktiv als der

Puffer lang ist, werden die restlichen aktiven Meldungen nicht in der Liste ausgegeben. Es wird diesbezüglich keine eigene Meldung erzeugt.

Puffer für MeldungsLog als String (Meldungshistorie String)

Im Puffer für das MeldungsLog im Format String werden alle aufgetretenen Meldungen angezeigt. Diese Daten werden als globale Daten abgelegt. Die Verwendung ist optional und kann bei der Konfiguration an- bzw. abgewählt werden.

Die String-Texte sind wahlweise in den Sprachen Deutsch, Englisch, Französisch und Italienisch auf der Steuerung hinterlegt. Weitere Sprachen (nur ASCII-Zeichencode) sind möglich und können vom Speichermedium des SIMOTION Gerätes ins System geladen werden. Detaillierte Informationen siehe Abschnitt Laden der Sprache vom Speichermedium des SIMOTION Gerätes (Seite 43).

Für jede Meldung werden folgende Informationen abgelegt:

- Information der aktiven Meldung
- Zeitstempel **Meldung gegangen**

Dieser Puffer befindet sich in der Programmunit **fLMsgHdl** und hat die Bezeichnung *gsLMsgHdlMessageLogString*.

- Pro Meldung werden im Puffer für das gesamte MeldungsLog String 314 Byte benötigt.
- Die Länge des Speichers für das gesamte MeldungsLog String ist über die Konstante LMSGHDL_LENGTH_OF_MESSAGE_LOG einstellbar. Die Voreinstellung liegt bei 200 Einträgen.
  Somit benötigt das gesamte MeldungsLog, Meldungshistorie als String ca. 62,8 kByte Speicher im globalen Datenbereich.
  Ausnahme bei SIMOTION D410, da beträgt die Voreinstellung 150 Einträge.

Der Puffer für MeldungsLog als Rohdaten ist als Ringpuffer aufgebaut.

Puffer für aktive Meldungen String

Im Puffer für aktive Meldungen werden alle noch nicht quittierten, bzw. nicht quittierbaren Meldungen angezeigt. Diese Daten werden als globale Daten abgelegt. Die Verwendung ist optional und kann bei Konfiguration an- bzw. abgewählt werden.

Die String-Texte sind wahlweise in den Sprachen Deutsch, Englisch, Französisch und Italienisch vorhanden. Weitere Sprachen (nur ASCII-Zeichencode) sind möglich und können vom Speichermedium des SIMOTION Gerätes ins System geladen werden. Detaillierte Informationen siehe Abschnitt Laden der Sprache vom Speichermedium des SIMOTION Gerätes (Seite 43).

Für jede Meldung werden folgende Informationen abgelegt:

- Kennung für Quelle der Meldung
- Stufe der Meldung (Fehler, Störung, Warnung)
- Quittierungsart der Meldung, z. B. benötigen manche DO-Meldungen ein PowerOn als Quittierung

- Sprachabhängiger Meldungstext inklusive Zusatzinformationen
  Hierdurch wird erreicht, dass alle Meldungen aus allen Quellen trotz unterschiedlicher Informationen die gleiche Form der Meldung besitzen.
- Kategorie der Meldung
- Zeitstempel **Meldung gekommen**

Dieser Puffer befindet sich in der Programmunit **pLMsgHdl** und hat die Bezeichnung *gsLMsgHdlActiveMessageString*.

- Pro Meldung werden im Puffer für die aktiven Meldungen 291 Byte benötigt.
- Die Länge des Speichers für die aktiven Meldungen ist über die Konstante LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES einstellbar. Die Voreinstellung liegt bei 100 Einträgen.
  Somit benötigt der gesamte Meldungspuffer für aktive Meldungen als Rohdaten ca. 29,1 kByte Speicher im globalen Datenbereich.

Der Puffer für aktive Meldungen ist kein Ringpuffer und wird nacheinander mit den gefundenen Einträgen im MeldungsLog beschrieben. Sind mehr Meldungen aktiv als der Puffer lang ist, werden die restlichen aktiven Meldungen nicht in der Liste ausgegeben. Es wird auch keine entsprechende Meldung erzeugt.

3.4.2.3 Funktionen zum Eintragen von anwenderdefinierten Meldungen

Mit den Funktionen **FCLMsgHdlWriteUserMessageToBuffer** und **FCLMsgHdlWriteFBFCMessageToBuffer** werden die anwenderdefinierten Meldungen ans Meldungshandling übergeben und die zugehörige Meldungsklasse übernommen. Des Weiteren wird mit Hilfe dieser Funktionen bei angewähltem AlarmS-Verfahren, bzw. Bitmeldeverfahren die entsprechende Meldung im System ausgelöst. Die Funktionen können in allen Tasks des Ablaufsystems aufgerufen werden.

Von dieser Funktion werden sowohl die AlarmS Meldungen, wie auch die entsprechenden Bits beim Bitmeldeverfahren gesetzt.

Die Texte für die anwenderdefinierten Meldungen müssen in allen gewünschten Sprachen vom Anwender selbst erstellt werden. Die Funktionsweise und Handhabung unterschiedlicher Sprachen ist wie bei der Pufferverwaltung.

Hinweis

Jede anwenderdefinierte Meldung kann nur einmalig im System aktiv sein. Ist eine Meldung bereits aktiv und wird erneut an das Meldungshandling übergeben, wird die Meldung nicht eingetragen. Erst nachdem eine anwenderdefinierte Meldung quittiert wurde, kann sie erneut ausgelöst werden.

Bei den anwenderdefinierten Meldungen gibt es zwei unterschiedliche Arten von Meldungen:

- anwenderdefinierte Meldungen der Applikation
- anwenderdefinierte Meldungen von Funktionen und Funktionsbausteinen

Siehe auch

Funktionen FCLMsgHdlWriteUserMessageToBuffer und FCLMsgHdlWriteFBFCMessageToBuffer (Seite 89)

3.4.2.4 AlarmS

AlarmS ist eine vorhandene Funktion und kann von der Applikation Meldungshandling genutzt werden. Der Anwender aktiviert dazu das AlarmS-Verfahren im SIMOTION Projekt. Ist AlarmS aktiviert, wird zu jeder anwenderdefinierten Meldung die entsprechend projektierte AlarmS-Meldung ausgelöst. Detaillierte Informationen dazu finden Sie im Abschnitt Einbettung AlarmS-Verfahren oder Bitmeldeverfahren (Seite 64).

3.4.2.5 Bitmeldeverfahren

Das Bitmeldeverfahren ist eine vorhandene Funktion und kann von der Applikation Meldungshandling genutzt werden. Der Anwender aktiviert dazu das Bitmeldeverfahren im SIMOTION Projekt. Detaillierte Informationen dazu finden Sie im Abschnitt Einbettung AlarmS-Verfahren oder Bitmeldeverfahren (Seite 64).

3.4.2.6 Verhalten bei Verarbeitungsfehlern in Programmen

Die Fehlerreaktion bei Programmfehlern wird durch das Konfigurations-Skript für alle Tasks auf die **ExecutionFaultTask** eingestellt. Tritt in einer sequenziellen Task ein Programmfehler, z. B. Zugriff außerhalb der Arraygrenzen auf, wird die ExecutionFaultTask gestartet. Die Task, in der dieser Fehler aufgetreten ist, wird abgebrochen, das SIMOTION Gerät bleibt im Betriebszustand RUN. Beim Durchlauf der ExecutionFaultTask werden die dabei ausgegebenen Meldungsinformationen in einen Zwischenspeicher des Meldungshandlings abgelegt. Dieser Zwischenpuffer liegt im netzausfesten Datenbereich (RETAIN). Da das SIMOTION Gerät anschließend automatisch in den Betriebszustand STOP geht, kann diese Meldung nicht mehr in den Meldepuffer eingetragen werden. Wird das SIMOTION Gerät wieder in den Betriebszustand RUN gestellt, wird die Meldung in den Meldungspuffer übernommen und erscheint in der Meldehistorie und den aktiven Meldungen. Diese Meldung kann dann durch Quittierung im Meldungshandling quittiert werden.

3.4.2.7 Hochlauf des Meldungshandlings

Beim Hochlauf des Meldungshandlings werden folgende Aktionen im Meldungshandling durchgeführt:

- Initialisierung des Meldungspuffers für die Meldungshistorie im Format Rohdaten. Alle beim Abschalten der Maschine aktiven Meldungen werden beim Hochlauf automatisch quittiert.
- Nach Durchlauf des Konfigurations-Skriptes werden der MeldungsLog im Format Rohdaten im netzausfesten Datenbereich (Retain) immer gelöscht. Werden Änderungen im Projekt durchgeführt, muss das Konfigurations-Skript erneut durchlaufen werden. Nachdem das Konfigurations-Skript beendet ist, kann nicht sichergestellt werden, dass die vom Konfigurations-Skript ermittelten projektierten Informationen zu den bereits vorhandenen Informationen im MeldungsLog Rohdaten zusammenpassen.

- Falls auf dem Speichermedium des SIMOTION Gerätes vorhanden, Einwechseln der im Meldungshandling hinterlegten Sprache. Diese Aktion wird nur durchgeführt, wenn im Meldungshandling das Format String angewählt wurde.
- Erzeugen einer neuen Meldung für den Neustart des SIMOTION Gerätes
- Überwachung des Hochlaufs aller am SIMOTION Gerät projektierten Devices. Fehlt ein projektiertes Device, bzw. ist ein Device nicht betriebsbereit, wird eine Meldung ins Meldungshandling eingetragen.
- Ermittlung von speziellen Informationen über alle projektierten Driveobjekte an SINAMICS Baugruppen.
- Auslesen der Namen von Antriebsobjekten für das Meldungshandling im Format String.
- Uhrzeitsynchronisation aller SINAMICS Baugruppen auf RTC (Real Time Clock) von SIMOTION (nur bei angewählter Uhrzeitsynchronisation und bei Verwendung eines Standardtelegrammes 39x an der Control Unit eines SINAMICS Objektes
- Wurde im Meldungshandling das MeldungsLog im Format String angewählt, werden die Meldungspuffer erzeugt.
- Tritt beim Hochlauf des Meldungshandlings ein Fehler in der Pufferverwaltung auf, könnte es vorkommen, dass der Hochlauf niemals beendet wird (Variable *boInitDriveReady* = TRUE). Aus diesem Grund wird die Hochlaufprüfung sofort beendet, wenn die Pufferverwaltung einen Fehler meldet. Dieser Fehler wird an das Meldungshandling übergeben und kann somit ausgegeben werden.

Hinweis

Bei Verwendung der Pufferverwaltung im Format String und einem großen Puffer für die Meldungshistorie kann die Erstellung der String-Puffer im Hochlauf sehr lange dauern. Um diesen Vorgang zu beschleunigen, wurde in der Unit **cPublic** der Bibliothek **LMsgHdl** die Konstante LMSGHDL_NUMBER_OF_STRING_MESSAGES_PER_CYCLE_IN_STARTUP angelegt. Abhängig von deren Wert, werden pro Zyklus in der BackgroundTask mehrere Strings hintereinander abgebildet. Dadurch wird die Zeit, bis der Hochlauf beendet ist, deutlich verkürzt.

Nach einem neuen Hochlauf des SIMOTION Gerätes dürfen keine alten Meldungen aktiv sein. Alle vor dem Hochlauf nicht gegangenen Meldungen werden beim Hochlauf automatisch quittiert. Nachdem alle Meldungen quittiert wurden, wird ein neuer Eintrag für den Hochlauf des SIMOTION Gerätes in den MeldungsLog eingetragen. Dieser Eintrag erhält die Zeitstempel **Zeit gekommen = Zeit gegangen = aktueller Wert der RTC**.

3.4.2.8 Quittieren der aktiven Meldungen

Das globale Quittieren aller Meldungen geschieht über die globale Variable *gboLMsgHdlMsgHdlGlobalAcknowledge* in der Programmunit **pLMsgHdl**. Die Variable muss durch die Applikation auf TRUE gesetzt werden. Durch die steigende Flanke wird im Meldungshandling ein globales Quittieren aller anstehenden Meldungen ausgelöst. Nach der Quittierung wird die Variable automatisch wieder auf FALSE gesetzt. Bei der Quittierung durch das Meldungshandling werden alle anstehenden Fehler und Meldungen an SIMOTION und SINAMICS quittiert. Bei aktivem AlarmS-Verfahren, bzw. Bitmeldeverfahren, werden diese aktiven Meldungen ebenfalls systemseitig quittiert. Alle aktiven Meldungen (außer Warnungen an Antriebsobjekten) werden im Meldungshandling zurückgesetzt. Stehen nach der Quittierung Fehler und Meldungen weiterhin an, werden diese neu ins Meldungshandling übernommen.

Hinweis

Alle Fehler und Meldungen, die in der Steuerung auftreten bzw. im Meldungshandling angezeigt werden, dürfen ausschließlich nur über das Meldungshandling quittiert werden. Wird eine Quittierung am Meldungshandling vorbei ausgelöst, kann dies nicht vom Meldungshandling erkannt werden. Somit wäre die Anzeige im Meldungshandling fehlerhaft.

3.4.2.9 Filtern von Meldungen an ein HMI / SIMOTION IT

Allgemeines

Die Filter werden an der Ausgabeschnittstelle zu SIMOTION IT / HMI eingesetzt. Hierzu wird die Implementierung innerhalb der FB **FBLMsgHdlActiveMsgBaseDataToHMI**, **FBLMsgHdlMsgLogBaseDataToHMI**, **FBLMsgHdlActiveMsgSgToHMI** und **FBLMsgHdlMsgLogSgToHMI** ausgeführt. Alle Meldungen werden in die jeweiligen globalen Gesamtpuffer eingetragen. Somit kann innerhalb der Ausgabe auf HMI / SIMOTION IT ausgewählt werden, welche Meldungsquellen angezeigt werden und welche nicht.

Meldungsquellen

Es werden folgende Meldungsquellen unterschieden:

- Meldungen von Technologieobjekten
  Bei Anwahl werden alle Meldungen (aktiv bzw. Meldungslog), die durch Technologieobjekte erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.
- DO-Alarm
  Bei Anwahl werden alle Alarme (aktiv bzw. Meldungslog), die durch Antriebsobjekte erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.
- DO-Warnungen
  Bei Anwahl werden alle Warnungen (aktiv bzw. Meldungslog), die durch Antriebsobjekte erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.
- DO-Safety-Meldungen
  Bei Anwahl werden alle Safety-Meldungen (aktiv bzw. Meldungslog), die durch Antriebsobjekte erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.

- Meldungen durch Peripheriebaugruppen
  Bei Anwahl werden alle Meldungen (aktiv bzw. Meldungslog), die durch Peripheriebaugruppen erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.
- Anwenderdefinierte Meldungen
  Bei Anwahl werden alle Meldungen (aktiv bzw. Meldungslog), die durch den Anwender erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.
- Systemmeldungen
  Bei Anwahl werden alle Meldungen (aktiv bzw. Meldungslog), die durch das System erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben. Hierzu zählen TimeFault-Meldungen, TimeFault-Meldungen der BackgroundTask und ExecutionFault-Meldungen.
- Meldungen durch Meldungshandling
  Bei Anwahl werden alle Meldungen (aktiv bzw. Meldungslog), die durch das Meldungshandling erzeugt wurden an ein HMI bzw. SIMOTION IT ausgegeben.

Wird eine der hier beschriebenen Quellen abgewählt, wird die Meldung selber zwar weiterhin in die entsprechenden Puffer eingetragen, erscheint jedoch nicht am Ausgang der oben beschriebenen FB zur Ausgabe an ein HMI bzw. SIMOTION IT. Diese Meldungen werden lediglich für die Ausgabe herausgefiltert.

Die Anwahl der Filter geschieht dabei in der Variablen *gsLMsgHdlFilterToHMI* der Programmunit **pLMsgHdl**. Diese Variable übergibt die Filterkriterien an alle Ausgabefunktionsbausteine für ein HMI bzw. SIMOTION IT.

Struktur der Variable gsLMsgHdlFilterToHMI

Die Variable *gsLMsgHdlFilterToHMI* ist vom Typ *sLMsgHdlFilterToHMIType* und hat folgenden Aufbau. Die Variable *gsLMsgHdlFilterToHMI* kann an die bereits existierenden HMI FB **FBLMsgHdlActiveMsgSgToHMI**, **FBLMsgHdlMsgLogSgToHMI**, **FBLMsgHdlActiveMsgBaseDataToHMI** und **FBLMsgHdlMsgLogBaseDataToHMI** übergeben werden. Die dort resultierenden Ausgaben werden mit den jeweiligen Filterinformationen berücksichtigt.

Tabelle 3- 7 Struktur von sLMsgHdlFilterToHMIType

Parameter	Datentyp	Initialwert	Beschreibung
boShowTOMessages	BOOL	TRUE	Bei TRUE werden alle Meldungen aller TO angezeigt, bei FALSE werden sie herausgefiltert.
boShowDOWarnings	BOOL	TRUE	Bei TRUE werden alle Warnungen aller DO angezeigt, bei FALSE werden sie herausgefiltert.
boShowDOAlarms	BOOL	TRUE	Bei TRUE werden alle Alarme aller Do angezeigt, bei FALSE werden sie her-ausgefiltert.
boShowDOSafetyMessages	BOOL	TRUE	Bei TRUE werden alle Safety-Meldungen aller DO angezeigt, bei FALSE werden sie herausgefiltert.
boShowPeripheralMessages	BOOL	TRUE	Bei TRUE werden alle Meldungen aller Peripheriebaugruppen angezeigt, bei FALSE werden sie herausgefiltert.

Parameter	Datentyp	Initialwert	Beschreibung
boShowSystemMessages	BOOL	TRUE	Bei TRUE werden alle Systemmeldungen angezeigt, bei FALSE werden sie herausgefiltert.
boShowUserDefinedMessages	BOOL	TRUE	Bei TRUE werden alle anwenderdefinierte Meldungen angezeigt, bei FALSE werden sie herausgefiltert.
boShowMessagesFromMsgHdl	BOOL	TRUE	Bei TRUE werden alle Meldungen die durch das Meldungshandling erzeugt wurden angezeigt, bei FALSE werden sie herausgefiltert.

3.4.2.10 Modulare Maschine

Allgemeines

Mit dem Meldungshandling ist es möglich Einträge von Meldungen in den Meldepuffer objektgranular zu unterdrücken.

Beispiel 1

Der Motor eines Transportbandes ist defekt. In das Meldungshandling werden bei Ausfall des TO bzw. DO, Fehler dieser Achse bzw. Antrieb eingetragen. Da der Antrieb für den Betrieb der Maschine nicht zwingend erforderlich ist, soll dieser aus dem Meldehandling und aus der Kundenapplikation abgewählt werden. Nach dem Setzen der Eigenschaft werden dann von diesen Objekten keine Fehler mehr in das Meldehandling eingetragen.

Beispiel 2

Teilinbetriebnahme einer Maschine

Die Hardware einer Maschine ist zu Beginn der Inbetriebnahme nicht vollständig, z. B. fehlen ein Motor und Peripherieteilnehmer. Einträge dieser Objekte in das Meldehandling sollen unterdrückt werden.

Unterdrückbare Meldungen an Objekten

Folgende Objekte können unterdrückt werden:

- Technologieobjekte
- Driveobjekte
- Peripheriebaugruppen

Jedem Objekt ist eine Einstellung zugeordnet, in dem festgelegt wird, ob das Meldungshandling das Objekt beobachten soll oder nicht. Die so festgelegten Einstellungen werden vom Meldungshandling spannungsausfallsicher gespeichert und liegen somit nach Power OFF / ON weiterhin an.

Für die Festlegung der zu überwachenden Objekte gibt es zwei unterschiedliche Varianten.

- Zunächst kann eine Grundfestlegung im Sourcecode des Meldungshandlings eingestellt werden. Dies geschieht in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlInitProjectInfo**.
- Darüber hinaus kann diese Basiseinstellung zur Laufzeit vom Anwender geändert bzw. angepasst werden. Diese zur Laufzeit geänderte Einstellung überschreibt die Basiseinstellung und ist anschließend bei jedem Neustart gültig. Die Basiseinstellungen werden vom Meldungshandling übernommen, wenn entweder das Konfigurations-Skript neu durchlaufen wurde, oder wenn der Anwender selber im Sourcecode der Progammunit **fLMsgHdlInit** die globale Konstante LMSGHDL_SCRIPT_COUNTER inkrementiert, neu übersetzt und das Projekt in das SIMOTION Gerät lädt.

Basiseinstellungen im Sourcecode

Die Basiseinstellungen im Sourcecode werden wie folgt durchgeführt:

- TO Achse mit DO
  Soll das TO überwacht werden, wird dies in **fLMsgHdlInit** in der Funktion **FCLMsgHdlInitProjectInfo** über *gasLMsgHdlaxes[numberOfAxis].boToUsedInProject := TRUE* übergeben. Das zugehörige DO wird über *gasLMsgHdlaxes[numberOfAxis].boRelatedDoUsed := TRUE* übergeben.
- Alle anderen TO
  Innerhalb der Übergabestruktur aller TO Typen gibt es eine entsprechende Einstellung, mit der die Überwachung für jedes TO eingeschaltet bzw. ausgeschaltet werden kann, z. B. *gasLMsgHdlExternalEncoders[numberOfTO].boToUsedInProject := TRUE.*
- DO mit zyklischer Kommunikation
  Innerhalb der Übergabestruktur aller DO mit zyklischer Kommunikation gibt es eine entsprechende Einstellung, mit der die Überwachung für jedes DO eingeschaltet bzw. ausgeschaltet werden kann, z. B. *gasLMsgHdlDOsCyclic[numberOfCyclicDO].boDoUsedInProject := TRUE.*

- DO ohne zyklische Kommunikation
  Innerhalb der Übergabestruktur aller DO ohne zyklische Kommunikation gibt es eine entsprechende Einstellung, mit der die Überwachung für jedes DO eingeschaltet bzw. ausgeschaltet werden kann, z. B.
  gasLMsgHdlDOsACyclic[numberOfCyclicDO].boDoUsedInProject := TRUE.
- Peripheriebaugruppen
  Innerhalb der Übergabestruktur aller Peripheriebaugruppen gibt es eine entsprechende Einstellung, mit der die Überwachung für jede Peripheriebaugruppe eingeschaltet bzw. ausgeschaltet werden kann, z. B.
  gasLMsgHdlPeripheralDevices[numberOfDevice].boUsedInProject := TRUE. Wird für eine Peripheriebaugruppe *boUsedInProject := FALSE* eingestellt, ist es zwingend notwendig, dass die Baugruppe auch tatsächlich nicht vorhanden ist. Ist sie dennoch vorhanden, wird ein Fehler des Meldungshandlings diesbezüglich ausgegeben. Bei Abwahl einer SINAMICS-Peripheriebaugruppe wird keine Uhrzeitsynchronisation auf dieser Baugruppe durchgeführt.

Hinweis

Ist eine Peripheriebaugruppe vom Typ SINAMICS nicht vorhanden, müssen alle zum Gerät gehörenden Objekte abgewählt werden. D. h. für alle TO, alle DO und die Peripheriebaugruppen selber muss die entsprechende Eigenschaft auf FALSE gesetzt werden. Wird dies nicht gemacht, kommt es zu Fehlermeldungen des Meldungshandlings.

Einstellungen zur Laufzeit

Die Einstellungen zur Laufzeit werden wie folgt durchgeführt:

Die hier beschriebenen Einstellungen werden innerhalb der Programmunit **pLMsgHdl** in Retaindaten abgelegt.

- TO Achse mit DO
  Die Variable *gsLMsgHdlMoMaTOAxis* ist vom Typ *sLMsgHdlMoMaTOAxisType* und befindet sich in der Programmunit **pLMsgHdl**. Dadurch sind die Einstellungen zum Thema modulare Maschine auch über den Symbolbrowser durchzuführen.
  Die Information, welche Einträge zu welchem Technologieobjekt gehören, werden durch das Meldungshandling übergeben.

Der Datentyp ist wie Folgt definiert:

Tabelle 3- 8 Struktur von sLMsgHdlMoMaTOAxisType

Parameter	Datentyp	Initialwert	Beschreibung
toReference	ANYOBJECT	TO#NIL	Die Achs-Referenz wird aus der Ablaufsoftware des Meldungshandlings heraus gesetzt.
sgToName	STRING		Name der Achse
boRelatedDOUsed	BOOL	TRUE	Bei FALSE wird das zum TO gehörende DO nicht vom Meldungshandling überwacht.
boTOUsedInProject	BOOL	TRUE	Bei FALSE wird das TO nicht vom Meldungshandling überwacht.

- Alle anderen TO
  Die Variablen *gsLMsgHdlMoMaxxx* (Typ des TO) sind vom Typ *sLMsgHdlMoMaTOType* und befinden sich in der Programmunit **pLMsgHdl**. Dadurch sind die Einstellungen zum Thema modulare Maschine auch über den Symbolbrowser durchzuführen.
  Die Information, welche Einträge zu welchem Technologieobjekt gehören, werden durch das Meldungshandling übergeben.
  Je nach aktivem Technologiepaket gibt es folgende Variablen:
  - *gasLMsgHdlMoMaExternalEncoders* für externe Geber
  - *gasLMsgHdlMoMaMeasuringInputs* für Messtaster
  - *gasLMSGHDLMoMaOutputCams* für Nocken
  - *gasLMsgHdlMoMaCamTracks* für Nockenspur
  - *gasLMsgHdlMoMaCams* für Kurvenscheibe
  - *gasLMsgHdlMoMaFollowingObjects* für Gleichlaufobjekt
  - *gasLMsgHdlMoMaPathObjects* für Bahnobjekt
  - *gasLMsgHdlMoMaFixedGears* für festes Getriebe
  - *gasLMsgHdlMoMaAdditionObjects* für Addierobjekt
  - *gasLMsgHdlMoMaFormulaObjects* für Formelobjekt
  - *gasLMsgHdlMoMaSensors* für Sensor
  - *gasLMsgHdlMoMaControllerObjects* für Reglerobjekt
  - *gasLMsgHdlMoMaTemperatureControllers* für Temperaturkanal

Der Datentyp ist wie Folgt definiert.

Tabelle 3- 9 Struktur von sLMsgHdlMoMaTOType

Parameter	Datentyp	Initialwert	Beschreibung
toReference	ANYOBJECT	TO#NIL	Die Achs-Referenz wird aus der Ablaufsoftware des Meldungshandlings heraus gesetzt.
sgToName	STRING		Name der Achse
boTOUsedInProject	BOOL	TRUE	Bei FALSE wird das TO nicht vom Meldungshandling überwacht.

- DO mit zyklischer Kommunikation
  Die Variable *gasLMsgHdlMoMaDosCyclic* ist vom Typ *sLMsgHdlMoMaDOsCyclicType* und befindet sich in der Programmunit **pLMsgHdl**. Dadurch sind die Einstellungen zum Thema modulare Maschine auch über den Symbolbrowser durchzuführen.
  Die Information, welche Einträge zu welchem Antriebsobjekt gehören, werden durch das Meldungshandling übergeben.

Der Datentyp ist wie Folgt definiert.

Tabelle 3- 10 Struktur von sLMsgHdlMoMaDOsCyclicType

Parameter	Datentyp	Initialwert	Beschreibung
sgDoName	STRING		Name des DO
i32LogAddress	DINT	0	logische Adresse des DO

Parameter	Datentyp	Initialwert	Beschreibung
eIoId	enumIoIdType	INPUT	Datenrichtung der logischen Adresse
boDOUsedInProject	BOOL	TRUE	Bei FALSE wird das DO nicht vom Meldungshandling überwacht.

- DO ohne zyklische Kommunikation
  Die Variable *gasLMsgHdlMoMaDosAcyclic* ist vom Typ *sLMsgHdlMoMaDOsACyclicType* und befindet sich in der Programmunit **pLMsgHdl**. Dadurch sind die Einstellungen zum Thema modulare Maschine auch über den Symbolbrowser durchzuführen. Die Information, welche Einträge zu welchem Antriebsobjekt gehören, werden durch das Meldungshandling übergeben.

Der Datentyp ist wie Folgt definiert.

Tabelle 3- 11 Struktur von sLMsgHdlMoMaDOsACyclicType

Parameter	Datentyp	Initialwert	Beschreibung
sgDoName	STRING		Name des DO
i32LogAddress	DINT	0	logische Adresse des DO
eIoId	enumIoIdType	INPUT	Datenrichtung der logischen Adresse
u8DoNumber	USINT	0	DO-Nummer
boDOUsedInProject	BOOL	TRUE	Bei FALSE wird das DO nicht vom Meldungshandling überwacht.

- Peripheriebaugruppen
  Die Variable *gasLMsgHdlMoMaPeripheralDevices* ist vom Typ *sLMsgHdlMoMaPeripheralDevicesType* und befindet sich in der Programmunit **pLMsgHdl**. Dadurch sind die Einstellungen zum Thema modulare Maschine auch über den Symbolbrowser durchzuführen. Die Information, welche Einträge zu welcher Peripheriebaugruppe gehören, werden durch das Meldungshandling übergeben.

Der Datentyp ist wie Folgt definiert.

Tabelle 3- 12 Struktur von sLMsgHdlMoMaPeripheralDevicesType

Parameter	Datentyp	Initialwert	Beschreibung
sgDeviceName	STRING		Name der Peripheriebaugruppe
u32MasterSystemId	UDINT	0	*masterSystemId* des Bussystems, an dem die Periepheriebaugruppe projektiert wurde.
u32SlaveAddress	UDINT	0	Slaveadresse der Peripheriebaugruppe
boUsedInProject	BOOL	TRUE	Bei FALSE wird die Peripheriebaugruppe nicht vom Meldungshandling überwacht. Sie darf nicht vorhanden sein und wird auch nicht uhrzeitsynchronisiert (SINAMICS-Baugruppe).

Die Übernahme bzw. Gültigkeit dieser Informationen wird über das Setzen von *gboLMsgHdlActivateNewMoMaData* in der Programmunit **pLMsgHdl** durchgeführt. Nach der Übernahme wird die Einstellung *gboLMsgHdlActivateNewMoMaData* automatisch zurückgenommen. Diese Einstellung ist anschließend nach jedem Neustart Power OFF / ON gültig.

3.4.2.11 DO-Safety-Meldungen

Allgemeines

Mit dem Meldungshandling ist es möglich, zu den DO-Fehlern und Warnungen auch aktive Safety-Meldungen im Meldungshandling anzeigen zu lassen. Damit an einem DO die Safety-Meldungen ausgelesen werden, muss in der Unit **fLMsgHdlInit** bei jedem DO mit aktiver Safety Projektierung die Zusatzinformation *eCheckSIMessages* mit dem Wert BY_DO_ADDRESS versorgt werden. Da heute kein Bit für aktive Safety-Meldungen in der zyklischen Kommunikation vorhanden ist, muss an einem DO mit aktivem Safety, über die azyklische Kommunikation, direkt im DO ausgelesen werden, ob Safety-Meldungen aktiv sind oder nicht. Da somit jedes Mal azyklisch ausgelesen werden muss, ob Safety-Meldungen aktiv sind oder nicht, sollte *eCheckSIMessages* auch nur bei den DOs gesetzt werden, bei denen Safety-Meldungen auftreten können. Bei allen anderen DO soll DISABLED eingestellt sein (default).

Tritt an einem DO eine Safety-Meldung auf, wird diese anschließend in die Meldungspuffer eingetragen. Dabei haben Safety-Meldungen annähernd das gleiche Verhalten wie Warnungen am DO. D. h., dass Safety-Meldungen nicht über das Meldungshandling quittiert werden können. Ist eine Safety-Meldung am DO nicht mehr aktiv, wird dies vom Meldungshandling erkannt und mit dem entsprechenden Zeitstempel im Meldungshandling automatisch quittiert.

Die Information, ob an einem DO Safety-Meldungen überwacht werden soll, kann bei den drei unterschiedlichen DO-Typen an folgenden Stellen aktiviert werden:

DO mit TO

```
gasLMsgHdlaxes[…].eCheckSIMessages := BY_DO_ADDRESS;
```

DO mit zyklischer Kommunikation

```
gasLMsgHdlDOsCyclic[…].eCheckSIMessages := BY_DO_ADDRESS;
```

DO ohne zyklische Kommunikation

```
gasLMsgHdlDOsAcyclic[…].eCheckSIMessages := BY_DO_ADDRESS;
```

Tabelle 3- 13 Enum für eLDPV1CheckSIMessagesType

Enum-Bezeichner	Beschreibung
DISABLED (0)	Kontrolle von Safety-Meldungen am DO abgeschaltet
BY_DO_ADDRESS (2)	Kontrolle, ob Safety-Meldungen über azyklische Leseauftrag aktiv sind

Hinweis

Bei Verwendung von Safety werden die Meldungen automatisch vom Meldungshandling erfasst.

Sie müssen die hier beschriebenen Einstellungen nicht selbst durchführen.

BY_DO_ADDRESS ist voreingestellt.

3.4.2.12 Speichern des Zwischenpuffers der ShutdownTask

Im Meldungshandling ist der Zwischenpuffer für anwenderdefinierte Meldungen innerhalb der ShutdownTask netzausfallsicher. Somit werden anwenderdefinierte Meldungen der ShutdownTask beim erneuten Anlauf der Maschine im Meldungshandling angezeigt.

3.4.2.13 Speichern der aktuellen MeldungsLogs in das SIMOTION Gerät

Speichern des MeldungsLog

Es gibt zwei unterschiedliche Möglichkeiten, das MeldungsLog auf das Speichermedium des SIMOTION Geräts zu speichern. Beide Möglichkeiten können zusammen verwendet werden.

Hinweis

Mit der Funktion *AutoSave* wird das MeldungsLog gespeichert, sobald der Puffer voll ist. Alle Meldungen, die zu diesem Zeitpunkt keinen Zeitstempel *gegangen* haben, werden ohne diesen Zeitstempel gespeichert.

Möglichkeit 1

Es ist möglich den aktuellen Puffer im Format Rohdaten und Format String auf das Speichermedium des SIMOTION Geräts zu speichern. Um den Puffer zu speichern, muss in der Programmunit **pLMsgHdl** die globale Variable *gboLMsgHdlStartWriteCompleteMessageLogToStorageMedium* auf TRUE gesetzt werden. Der Name der Datei, in die die Informationen geschrieben werden, wird über die Variable *gu32LMsgHdlDataSetNoForExportMessageLog* gesetzt. (Default ist 0) Durch die steigende Flanke werden die aktuellen Meldepuffer auf das Speichermedium des SIMOTION Geräts gespeichert. Zusätzlich befinden sich in der Datei auch alle Informationen die benötigt werden, um das MeldungsLog im Format Rohdaten interpretieren zu können. Nach dem Speichern wird die globale Variable wieder auf FALSE gesetzt. Nach dieser Aktion befinden sich die Daten des Puffers auf dem Speichermedium des SIMOTION Geräts in folgendem Verzeichnis:

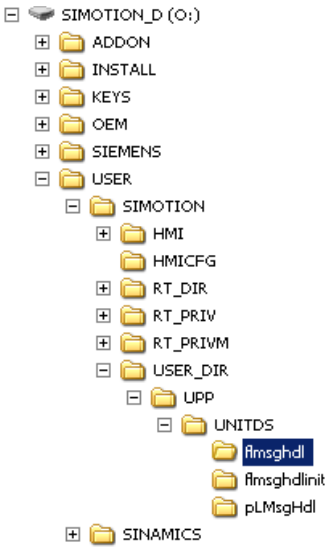


Bild 3-2 Verzeichnis fLMsgHdl

Im Verzeichnis **fLMsgHdl** hat die Datei mit den Pufferabbildern die Bezeichnung *dsxxxxxx.dat*, wobei *xxxxxx* die Zahl der in *gu32LMsgHdlDatasetNoForExportMessageLog* entspricht. Anhand dieser Datei ist z. B. eine Diagnose möglich.

Möglichkeit 2

Die Fehlerhistorie kann permanent auf das Speichermedium des SIMOTION Geräts gespeichert werden. Wenn der Log-Puffer im SIMOTION Gerät voll ist, wird automatisch gesichert.

Die Anzahl der Datensätze, die zur Sicherung der Daten verwendet werden, gibt der Anwender über Konstante vor LMSGHDL_MAX_NUMBER_OF_DATASETS_ON_STORAGE_MEDIUM in der Unit **cPublic** an. Das Anlegen der Datensätze erfolgt nach dem Prinzip eines Ringpuffers. Über LMSGHDL_AUTO_SAVE_MESSAGE_BUFFER_TO_STORAGE_MEDIUM in de Unit **cPublic** kann die Funktionalität eingeschaltet bzw. ausgeschaltet werden.

Eigenschaften der Funktionalität:

Das automatische Speichern wird über Konstanten des Meldungshandlings in der Unit **cPublic** aktiviert. Der Anwender gibt die Anzahl der Datensätze bzw. der Ringpuffergröße ebenfalls in der Unit **cPublic** an, z. B. 3.

Steht im Ringpuffer des Meldungshandlings ein Überlauf an, wird der Puffer datenkonsistent auf das Speichermedium des SIMOTION Geräts gespeichert. Die dort hinterlegten Informationen bekommen die Datensatznummern 1000 / 1001 / 1002 usw. Ist die maximale Anzahl an Datensätzen erreicht, wird im Ringpufferverfahren der erste Datensatz wieder überschrieben.

3.4.2.14 Laden der Sprache vom Speichermedium des SIMOTION Gerätes

Das Laden unterschiedlicher Sprachen geschieht wie folgt:

Der in der Unit **cPublic** der Bibliothek **LMsgHdl** gesetzte Wert der Konstanten LMSGHDL_LANGUAGE_FOR_MESSAGE_STRING gibt vor, mit welcher Sprache das Meldungshandling nach dem Hochlauf startet. Der Wert für die jeweilige Sprache entspricht dabei der Language-ID aus der STEP 7-Notation. Derzeit sind die Sprachen Deutsch, Englisch, Französisch und Italienisch integriert. Andere Sprachen müssen vom Anwender selbst erzeugt und geladen werden. Bei jedem Hochlauf des SIMOTION Gerätes lädt das Meldungshandling die entsprechenden Sprachdateien für Systemmeldungen und anwenderdefinierte Meldungen vom Speichermedium des SIMOTION Gerätes ins Meldungshandling. Sind diese Sprachdateien nicht vorhanden, wird die über das Konfigurations-Skript eingestellte Sprache (Deutsch oder Englisch) vom Meldungshandling verwendet.

Während des Betriebs des Meldungshandlings kann die aktive Sprachauswahl zwischen den auf dem Speichermedium des SIMOTION Gerätes hinterlegten Sprachen beliebig gewechselt werden. Dies geschieht über die Variablen *gu8LMsgHdlActiveLanguage* und *gboLMsgHdlStartChangeLanguage* aus der Unit **pLMsgHdl**. Hier wird die zu verwendende Sprache als Language-ID an die Variable *gu8LMsgHdlActiveLanguage* übergeben und mit einer steigende Flanke in der Variablen *gboLMsgHdlStartChangeLanguage* ausgetauscht. Sind die entsprechenden Sprachdateien auf dem Speichermedium des SIMOTION Gerätes vorhanden, werden nach dem Laden der Sprachmeldungen alle Meldungspuffer im Format String neu erstellt. Anschließend werden die Meldungspuffer direkt in der geänderten Sprache ausgegeben. Tritt beim Laden ein Fehler auf, weil z. B. die ausgewählte Sprache nicht auf dem Speichermedium des SIMOTION Gerätes vorhanden ist, wird eine Meldung ins Meldungshandling eingetragen.

Die Dateien mit den Systemmeldungen in den unterschiedlichen Sprachen müssen auf dem Speichermedium des SIMOTION Gerätes in folgendem Verzeichnis abgelegt werden: USER/SIMOTION/USER_DIR/UPP/UNITDS/pLMsgHdl

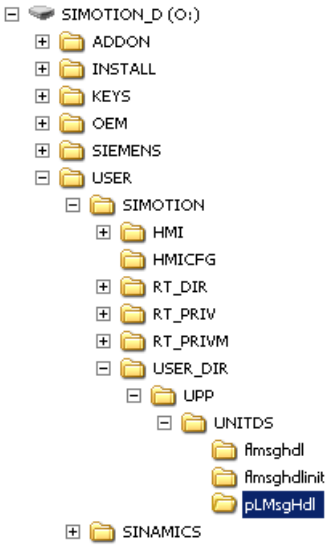


Bild 3-3 Verzeichnis für Sprachdateien

Bild 3-4 Sprachdateien

Die Namen der Dateien für die unterschiedlichen Sprachen ergeben sich dabei aus dem Kürzel **ds** inklusive der Language-ID aus der STEP 7-Notation für die jeweilige Sprache. Demnach beinhaltet die oben dargestellte Datei *ds000007.dat* im Verzeichnis **pLMsgHdl** alle systemseitigen Meldungstexte in Deutsch.

Die Dateien mit den anwenderdefinierten Meldungen in den unterschiedlichen Sprachen müssen auf dem Speichermedium des SIMOTION Gerätes in folgendem Verzeichnis abgelegt werden: USER/SIMOTION/USER_DIR/UPP/UNITDS/fLMsgHdlInit

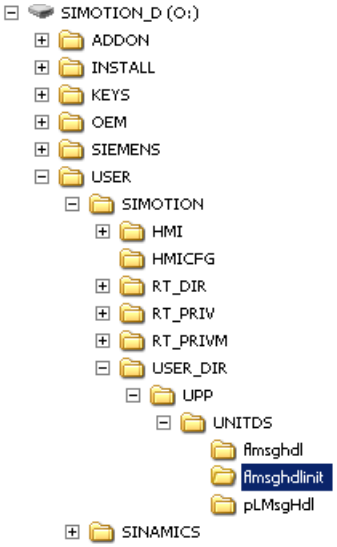


Bild 3-5 Verzeichnis für Sprachen der anwenderdefinierten Meldungen

Bild 3-6 Dateien für Sprachen der anwenderdefinierten Meldungen

Die Namen der Dateien für die unterschiedlichen Sprachen ergeben sich auch hier aus der Abkürzung **ds** und der Language-ID aus der STEP 7-Notation für die jeweilige Sprache. Demnach beinhaltet die oben dargestellte Datei *ds000007.dat* im Verzeichnis **fLMsgHdlInit** alle anwenderdefinierten Meldungstexte in Deutsch.

Tabelle 3- 14 Verwendete Sprach-Codes nach STEP 7-Notation (Language-ID)

Sprache	Wert [USINT]
Kein MeldungsLog im Format STRING	0
deutsch	7
englisch	9
spanisch	10
französisch	12
italienisch	16

3.4.2.15 Einzelquittierung

Einzelquittierung von Meldungen

Allgemeines

Neben der globalen Quittierung aller aktiven Meldungen können, je nach Meldungsquelle, alle aktiven Meldungen einer Quelle oder sogar einzelne Meldungen quittiert werden.

In der Programmunit **pLMsgHdl** wird über die Variable *gi32LMsgHdlNumberOfMessageInLog* dem Meldungshandling mitgeteilt, welche Meldung quittiert werden soll. Die Quittierung selbst wird wie bei der globalen Quittierung über *gboLMsgHdlGlobalAcknowledge* gestartet.

Sollen global alle aktiven Meldungen quittiert werden, muss in *gi32LMsgHdlNumberOfMessageInLog* eine -1 eingetragen sein (default).

Bei Einzelquittierung ergibt sich die Nummer der zu quittierenden Meldung aus der Nummer der Meldung im Puffer für aktive Meldungen. D. h., soll die erste aktive Meldung aus dem Meldungspuffer quittiert werden, muss vor dem starten der Quittierung in *gi32LMsgHdlNumberOfMessageInLog* die Nummer 1 eingetragen werden. Nach der Quittierung wird der Inhalt dieser Variable wieder auf -1 und *gboLMsgHdlGlobalAcknowledge* auf FALSE zurückgesetzt.

Einzelquittierung für die Meldungsquellen

- TO-Meldungen
  Einzelquittierung einer TO-Meldung heruntergebrochen bis auf gesamtes TO, an dem

Fehler anstehen, z. B. *_resetAxisError(ALL_ERRORS)*. Es werden, bezogen auf das ausgewählte TO, alle anstehenden Meldungen quittiert.

- DO-Meldungen
  Es können nur DO-Fehler quittiert werden. DO-Warnungen bzw. Safety-Meldungen lassen sich nicht quittieren, sondern gehen selbstständig sobald der Grund der Meldung nicht mehr vorliegt. Soll ein Fehler an einem DO quittiert werden, wird eine Einzelquittierung des betroffenen DO ausgelöst. Alle weiteren zu diesem DO gehörenden Fehler werden somit ebenfalls quittiert. Liegen die Fehler nach der Quittierung weiterhin an, werden diese erneut ins Meldungshandling eingetragen.
- Peripherie-Meldungen
  Nur die angewählte Peripheriemeldung wird quittiert. Die Peripheriemeldungen mit der Kennung 202, 204, 210 und 215 können nicht quittiert werden. Diese Meldungen werden bei der entsprechenden Gegenmeldung automatisch quittiert.
- Time-Fault Meldungen
  Nur die angewählte Time-Fault-Meldung wird quittiert.
- Meldungen durch Meldungshandling
  Nur die angewählte Meldung durch das Meldungshandling wird quittiert
- Anwenderdefinierte Meldungen
  Jede anwenderdefinierte Meldung wird einzeln quittiert. Die einzelne Quittierung hat ebenfalls Einfluss auf das Bitmeldeverfahren und AlarmS.

Auswahl der zu quittierenden Meldung über SIMOTION IT

Auf der SIMOTION IT Seite können einzelne, aktive Meldungen vom Anwender ausgewählt werden, die quittiert werden sollen. Durch die Verwendung von Schaltflächen für die Einzelquittierung wird sichergestellt, dass bei einer Einzelquittierung auch tatsächlich nur eine Meldung bzw. Meldungsquelle quittiert wird. Nachdem die Auswahl getroffen wurde, wird dem Meldungshandling beim Betätigen der zur Meldung gehörenden Schaltfläche **Acknowledge** diese Auswahl übertragen. Das Meldungshandling führt anschließend diese Einzelquittierung durch. Die Information die das Meldungshandling benötigt steht in *ai32NumberOfMessageInLog[LineInHMI - 1*] in der Variablen *gsLMsgHdlActiveMsgToHMI* der Unit **pLMsgHdl**. Diese Information wird bei Einzelquittierung an *gi32LMsgHdlNumberOfMessageInLog* der Unit **pLMsgHdl** übertragen. Gleichzeitig bekommt *gboLMsgHdlGlobalAcknowledge* eine steigende Flanke durch SIMOTION IT.

Anwenderdefinierte Meldung anhand der Meldungsnummer

Um eine Anwenderdefinierte Meldung "einzeln" quittieren zu können, wurde in der Programmunit **pLMsgHdl** eine neue globale Variable zur Übergabe der Meldungsnummer eingefügt.

Die Variable zur Übergabe der Nummer ist *gi32LMsgHdlNumberOfUserMessage*.

Um eine spezielle anwenderdefinierte Meldung zu quittieren muß die Nummer der Meldung in diese Variable eingetragen werden. Anschließend wird mit der steigenden Flanke an *gboLMsgHdlGlobalAcknowledge* die Quttierung ausgeführt.

Da die Anwenderdefinierten Meldungen zusätzlich das Bitmeldeverfahren und AlarmS unterstützen, werden bei der Einzelquittierung die Meldungen dieser Verfahren ebenfalls quittiert.

Über die Angabe der Meldungsnummer ist es somit möglich anwenderdefinierte Meldungen, anwenderdefinierte Meldungen von FBs/FCs, sowie Meldungen durch das Meldungshandling selber "gezielt" zu quittieren.

Da die anwenderdefinierte Meldung mit der Meldungsnummer 0 mehrfach aktiv sein kann, ist hier zu beachten, dass bei der Einzelquittierung nur der jeweils erste gefundene Eintrag mit der Kennung 0 quittiert wird. Sollen jedoch alle Meldungen mit der Kennung 0 gleichzeitig quittiert werden, so ist das Quittierungsverfahren unter Punkt 1.2 zu verwenden.

Hinweis

Es wird empfohlen, auf andere Einzelquittiermechanismen (Seite 47) zurückzugreifen, da es bei dem hier beschriebenen Verfahren zwischen Auswahl der Meldungsnummer und Auslösen der Quittierung zu Verschiebungen im Puffer kommen kann, so dass infolge dessen nicht die richtige Meldung quittiert wird.

Sammelquittierung von bestimmten Quellen

Um alle Meldungen einer bestimmten Quelle, z. B. eines bestimmten DOs, quittieren zu können wurde in der Programmunit **pLMsgHdl** die Variable *gsLMsgHdlAcknMessageInfo* vom Typ *sLMsgHdlAcknMessageInfoType* eingefügt. Somit ist es möglich alle Meldungen, die den dort möglichen Kriterien entsprechen, gleichzeitig zu quittieren.

Die Möglichkeiten dieser Quittierungsart werden im Folgenden näher beschrieben.

Tabelle 3- 15 Elemente der Struktur *sLMsgHdlAcknMessageInfoType*

Element	Datentyp	Initialisierungswert	Bedeutung
u8MessageSource	USINT	0	Quelle der zu quittierenden Meldungen im Meldungshandling
u16Parameter1	UINT	0	Wert zur Identifikation der zu quittierenden Meldung 1. (siehe Anhang B.1 zur Interpretation der Rohdaten)
i16Parameter2	INT	0	Wert zur Identifikation der zu quittierenden Meldung 2. (siehe Anhang B.1 zur Interpretation der Rohdaten)
b32Parameter3	REAL	16#FFFF_FFFF	Wert zur Identifikation der zu quittierenden Meldung 3. (siehe Anhang B.1 zur Interpretation der Rohdaten)
b32Parameter4	REAL	255	Wert zur Identifikation der zu quittierenden Meldung 4. (siehe Anhang B.1 zur Interpretation der Rohdaten)
b32Parameter10	REAL	0	Wert zur Identifikation der zu quittierenden Meldung 10. (siehe Anhang B.1 zur Interpretation der Rohdaten)

Die Elemente der Struktur zur Übergabe der Quittierungsinformationen entsprechen den Daten die durch die unterschiedlichen Meldungstypen im Rohdatenpuffer des Meldungshandlings eingetragen werden. Nähere Informationen befinden sich im Anhang B.1 der Dokumentation des Meldungshandlings.

Hinweis

Das hier beschriebene Quittierungsverfahren funktioniert prinzipiell nach der Methode dass die in der Quittierungs-Struktur geforderten Informationen 1:1 aus dem Rohdatenpuffer übernommen werden sollten.

Meldungen von Technologieobjekten

Hiermit können alle anstehenden Fehler eines einzelnen Technologieobjekts quittiert werden. Handelt es sich bei dem Technologieobjekt um eine Achse mit realem Antrieb, wird zusätzlich noch das unterlagerte Antriebsobjekt mit quittiert.

Um ein spezielles Technologieobjekt zu quittieren, müssen folgende Angaben an *gsLMsgHdlAcknMessageInfo* übergeben werden.

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 1;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= Kennung des entsprechenden TO-Typs
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= TO-Nummer
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= nicht relevant

Die TO-Nummer entspricht dem jeweiligen Subindex unter dem das Technologieobjekt in **fLMsgHdlInit** durch das Skript eingetragen wurde.

Die jeweiligen Kennungen für die TO-Typen sind im Anhang (Seite 126)hinterlegt.

Fehler an Antriebsobjekten

Hiermit können alle anstehenden Fehler eines einzelnen Antriebsobjekts (DO) quittiert werden.

Um ein spezielles Antriebsobjekt zu quittieren, müssen folgende Angaben an *gsLMsgHdlAcknMessageInfo* übergeben werden.

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= TO-Nummer (bei DO mit TO)
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= IoId (INPUT/OUTPUT)
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= logische Adresse zum DO
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= DO-Nummer
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= nicht relevant

Hierbei ist auf folgendes zu achten:

Handelt es sich beim Antriebsobjekt um einen realen Antrieb der einer Achse zugeordnet ist, muß zunächst in Parameter1 der Subindex der Achse eingetragen werden. Die Parameter 2-4 **müssen** auf Default-Werten stehen. Parameter10 ist nicht relevant.

Beispiel:

Quittierung des Antriebsobjekts zur Achse

gasLMsgHdlAxes[1].toReference := Folienabzug;

gsLMsgHdlAcknMessageInfo.au8MessageSource := 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1 := 1
gsLMsgHdlAcknMessageInfo.i16Parameter2 := 0
gsLMsgHdlAcknMessageInfo.b32Parameter3 := 16#FFFF_FFFF
gsLMsgHdlAcknMessageInfo.b32Parameter4 := 255
gsLMsgHdlAcknMessageInfo.b32Parameter10 := nicht relevant

Handelt es sich beim Antriebsobjekt um ein DO mit zyklischer Kommunikation, muß in Parameter1 der Defaultwert 0 eingetragen werden. Die Parameter 2-3 **müssen** die Werte für die IoId und die zugehörige logische Adresse des Telegramms bekommen. Parameter4 hat die Do-Nummer 255 und Parameter10 ist nicht relevant.

Beispiel:

Quittierung der CU mit Telegramm 390
gasLMsgHdlDOsCyclic[0].i32LogAddress := 256;
gasLMsgHdlDOsCyclic[0].eIOID := OUTPUT;

gsLMsgHdlAcknMessageInfo.au8MessageSource := 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1 := 0
gsLMsgHdlAcknMessageInfo.i16Parameter2 := 1 (0 Input / 1 Output)
gsLMsgHdlAcknMessageInfo.b32Parameter3 := 256
gsLMsgHdlAcknMessageInfo.b32Parameter4 := 255
gsLMsgHdlAcknMessageInfo.b32Parameter10 := nicht relevant

Handelt es sich beim Antriebsobjekt um ein DO ohne zyklische Kommunikation, muß in Parameter1 der Defaultwert 0 eingetragen werden. Die Parameter 2-4 **müssen** die Werte für die IoId, eine zugehörige logische Adresse und die DO-Nummer bekommen. Parameter10 ist nicht relevant.

Beispiel:

Quittierung des DO der TB30 ohne eigenes Telegramm
gasLMsgHdlDOsAcyclic[0].i32LogAddress := 16380; // diagnostic address
gasLMsgHdlDOsAcyclic[0].eIOID := INPUT;
gasLMsgHdlDOsAcyclic[0].u8DoNumber := 4;

gsLMsgHdlAcknMessageInfo.au8MessageSource := 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1 := 0
gsLMsgHdlAcknMessageInfo.i16Parameter2 := 0 (0 Input / 1 Output)
gsLMsgHdlAcknMessageInfo.b32Parameter3 := 16380
gsLMsgHdlAcknMessageInfo.b32Parameter4 := 4
gsLMsgHdlAcknMessageInfo.b32Parameter10 := nicht relevant

Meldungen von Peripheriebaugruppen

Mit der Übergabe von *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 4 werden alle aktiven, quittierbaren Meldungen durch Peripheriebaugruppen zurückgenommen.

Die Meldungen von Peripheriebaugruppen mit der Interrupt-Id 202, 204, 210 und 215 können nicht quittiert werden. Diese "verschwinden" von alleine, sobald die zugehörige Peripheriemeldung, dass der Fehler nicht mehr ansteht, erkannt wurde.

Meldungen durch Zeitüberlauf

Mit der Übergabe von *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 5 werden alle aktiven Meldungen, die durch Zeitüberlauf in Timer-Tasks bzw. der Background-Task erzeugt wurden quittiert.

Meldungen der Execution-Fault-Task nach Neuanlauf der Steuerung

Mit der Übergabe von *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 6 werden alle aktiven Meldungen, die durch Aufruf der Execution-Fault-Task erzeugt wurden quittiert.

Anwenderdefinierte Meldungen

Anwenderdefinierte Meldungen können wie folgt einzeln quittiert werden:

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 9;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= nicht relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= Meldungsnummer die quittiert werden soll
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= nicht relevant

Wird in Parameter3 die Meldungsnummer 0 übergeben, werden alle aktiven Meldungen mit der Meldungsnummer 0 quittiert.

Bei der Quittierung von anwenderdefinierten Meldungen werden, wenn verwendet, auch die zugehörigen Bitmeldungen bzw AlarmS-Meldungen quittiert.

Meldungen von FBs/FCs

Anwenderdefinierte Meldungen durch FBs/FCs können wie folgt einzeln quittiert werden.

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 8;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= nicht relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= nicht relevant

gsLMsgHdlAcknMessageInfo.b32Parameter3	:= Meldungsnummer die quittiert werden soll
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= nicht relevant

Bei der Quittierung von anwenderdefinierten Meldungen werden, wenn verwendet, auch die zugehörigen Bitmeldungen bzw AlarmS-Meldungen quittiert.

Darüber hinaus ist es auch noch möglich alle aktiven Meldungen durch FBs/FCs mit der gleichen functioBlockId gleichzeitig zu quittieren. Dabei werden alle Meldungen unabhängig von der Meldungsnummer in Abhängigkeit der functionBlockId im Übergabeparameter Parameter10 quittiert.

gsLMsgHdlAcknMessageIn-fo.au8MessageSource	:= 8;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= nicht relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= nicht relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= 5

Somit würden alle Meldungen durch FBs/FCs mit der Kennung 5 in Parameter10 (functionBlockId) gleichzeitig quittiert.

Meldungen durch das Meldungshandling

Mit der Übergabe von *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 10
werden alle aktiven Meldungen, die durch das Meldungshandling selber erzeugt wurden quittiert.

3.4.2.16 Gemeinsamer Puffer für Meldung gekommen / gegangen

Funktionalität

Um den neuen Puffer im Format Rohdaten nutzen zu können, muss zunächst die Programmunit **dLMsgHdl** in das SIMOTION Projekt importiert werden. Dieser Import wird heute nicht durch das Skript unterstützt und muss über den XML-Import einer Programmunit selber durchgeführt werden.

Nach dem Import muss noch das auskommentierte Define LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED in **dLMsgHdl** einkommentiert werden. Dieses Define muss ebenfalls in der Bibliotheksunit **cPublic** der Bibliothek **LMsgHdl** einkommentiert werden.

Nur wenn das Define in beiden Units einkommentiert wurde und nach einer anschließenden neuen Übersetzung des Projekts ist der neue Meldungspuffer aktiv und kann verwendet werden.

Zusätzlich muss in der Programmunit **fLMsgHdlInit** die Codezeile *USES dLMsgHdl* ein- und die Codezeile *USELIB LMsgHdl* auskommentiert werden.

Über die Konstante *LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED* wird bestimmt, wie viele Einträge im Puffer aufgenommen werden können. Da es sich bei diesem Puffer um einen Ringpuffer handelt, wird bei Überlauf damit begonnen die jeweils älteste Meldung zu überschreiben.

Die Konstante befindet sich in der Bibliotheksunit **cPublic** und muss vom Anwender auf den gewünschten Wert gesetzt werden. Default steht die Konstante auf 200.

Soll auf diesen Puffer per HMI zugegriffen werden, ist ebenfalls darauf zu achten, dass die Größe des Puffers 64k nicht überschreitet.

Aufbau des Puffers

Tabelle 3- 16 Elemente der Struktur *sLMsgHdlMessageLogBaseDataGoneAndOccurredType*

Element	Datentyp	Bedeutung
i16ActualIndex	INT	Aktueller Index im globalen Meldungs-Log für Rohdaten
au8MessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array für Information über Quelle der Meldung
au8MessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array für Information über Stufe der Meldung (Störung, Warnung, Fehler, Information)
au8AcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array für Information über die Quittierungsart der Meldung
au8ErrorClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array für Information über Fehlerklasse einer Meldung
aboOccuredMessage	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF BOOL	Array für Informationen, ob es sich bei der Meldung um eine gekommene oder gegangene Meldung handelt. TRUE zeigt an, dass es sich um eine gekommene Meldung handelt.
au16Parameter1	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF UINT	Array für Information Variable1 des jeweiligen Meldungstyps
ai16Parameter2	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF INT	Array für Information Variable2 des jeweiligen Meldungstyps
ab32Parameter3	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable3 des jeweiligen Meldungstyps
ab32Parameter4	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable4 des jeweiligen Meldungstyps
ab32Parameter5	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable5 des jeweiligen Meldungstyps
ab32Parameter6	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable6 des jeweiligen Meldungstyps
ab32Parameter7	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable7 des jeweiligen Meldungstyps
ab32Parameter8	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable8 des jeweiligen Meldungstyps
ab32Parameter9	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable9 des jeweiligen Meldungstyps

Element	Datentyp	Bedeutung
ab32Parameter10	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable10 des jeweiligen Meldungstyps
ab32Parameter11	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array für Information Variable11 des jeweiligen Meldungstyps
adtMessageGoneOccured	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DT	Zeitstempel, wann die entsprechende Meldung aufgetreten bzw. gegangen ist.
dLMsgHdl	LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED	alternative Meldungsvariante (standardmäßig nicht aktiviert)

Über die Konstante *LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED* wird bestimmt, wie viele Einträge im Puffer aufgenommen werden können. Da es sich bei diesem Puffer um einen Ringpuffer handelt, wird bei Überlauf damit begonnen die jeweils älteste Meldung zu überschreiben.

Die Konstante befindet sich in der Bibliotheksunit **cPublic** und muß vom Anwender auf den gewünschten Wert gesetzt werden. Default steht die Konstante auf 200.

Soll auf diesen Puffer per HMI zugegriffen werden, ist ebenfalls darauf zu achten, dass die Größe des Puffers 64k nicht überschreitet.

Integration

4

4.1 Erforderliche Technologieobjekte

Das Meldungshandling benötigt als Mindestvoraussetzung das Technologiepaket TP Cam. Je nach verwendeten Technologieobjekten kann das Meldungshandling aber auch mit den Technologiepaketen TP Path, TP Cam_ext und TControl betrieben werden.

4.2 Integration ins SIMOTION Projekt

4.2.1 Integration der Applikation in ein SIMOTION Projekt

Die Bibliotheken **LDPV1** und **LMsgHdl** können über den **Projektgenerator**, der im Lieferumfang von SIMOTION SCOUT (auf dem Speichermedium Utilities & Applications) enthalten ist, in das vorhandene Projekt integriert werden.

Um den Projektgenerator auszuführen, gehen Sie wie folgt vor:

1. Schließen Sie SMOTION SCOUT.
2. Doppelklicken Sie auf die Datei **ProjectGenerator.exe**.
3. Bestätigen Sie den Haftungsausschluss und wählen Sie die Option **Create a new project**.

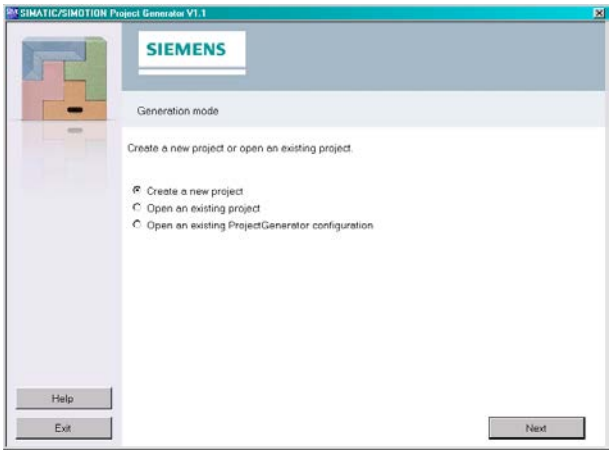


Bild 4-1 Projekt anlegen oder wählen

4. Wenn Sie **Create a new project** ausgewählt haben, geben Sie einen Projektnamen und den Ablageort des Projekts ein (der Pfad ist auch über **Browse Path** auswählbar) und klicken auf die Schaltfläche **Next**.

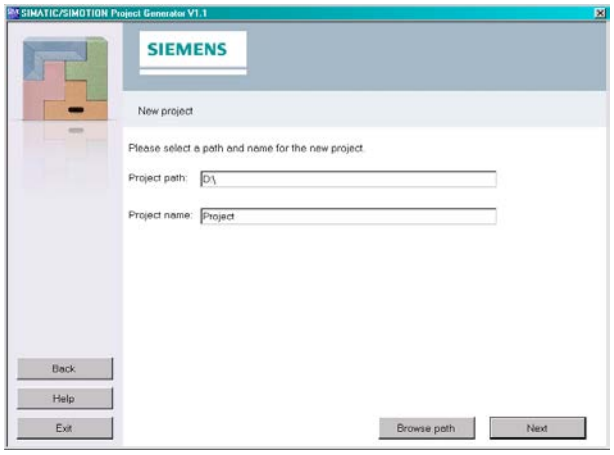


Bild 4-2 Projektname und Ablagepfad des Projekts

Das Fenster *SIMATIC/SIMOTION project data - device selection* wird geöffnet.

5. Wählen Sie im Fenster *SIMATIC/SIMOTION project data - device* selection unter **Select device category**, welches Gerät/welche Geräte in das Projekt integriert werden sollen:

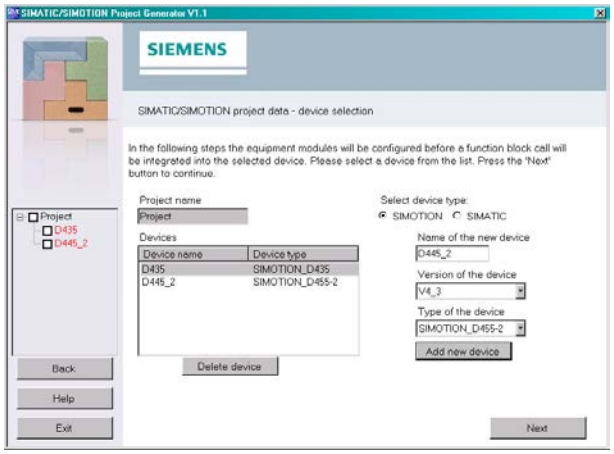


Bild 4-3 Gerät anlegen oder wählen

Abhängig von der Software Installation können Sie entweder nur SIMATIC Geräte oder SIMOTION und SIMATIC Geräte gemischt nacheinander auswählen.
Ist noch kein Gerät angelegt, können Sie im rechten Teil des Fensters ein neues Gerät anlegen. Geben Sie dazu den *Gerätenamen*, die *Version* und den *Typnamen* des Geräts ein und klicken Sie auf die Schaltfläche **Add new device**. Das neue Gerät wird in die Liste **Devices** übernommen.
Wenn Sie ein weiteres Gerät anlegen möchten, wiederholen Sie den Vorgang.

6. Markieren Sie in der Liste **Devices** das Gerät, das Sie konfigurieren möchten und klicken Sie auf die Schaltfläche **Next**.
Es öffnet sich das Fenster *SIMATIC/SIMOTION project data - equipment module selection*.

7. Wählen Sie im Fenster *SIMATIC/SIMOTION project data - equipment module selection* das oder die Standardmodule, die Sie in das gewählte Gerät integrieren möchten, in diesem Fall **Message Handling**, und klicken Sie auf die Schaltfläche **Next**.

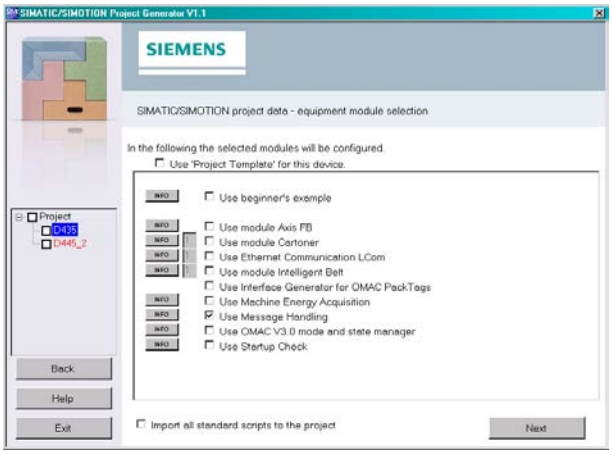


Bild 4-4 Modulauswahl

Wenn Sie auf die Schaltfläche **INFO** klicken, wird die Dokumentation zum jeweiligen Standardmodul geöffnet.
Bei einigen Standardmodulen, können Sie links die Anzahl der Module eingeben.

8. Konfigurieren Sie im Fenster *Message Handling – Configuration* den Aufruf des Funktionsbausteins mit den dazu notwendigen Datenbausteinen und Kommunikationsparametern und klicken Sie auf die Schaltfläche **Next**.

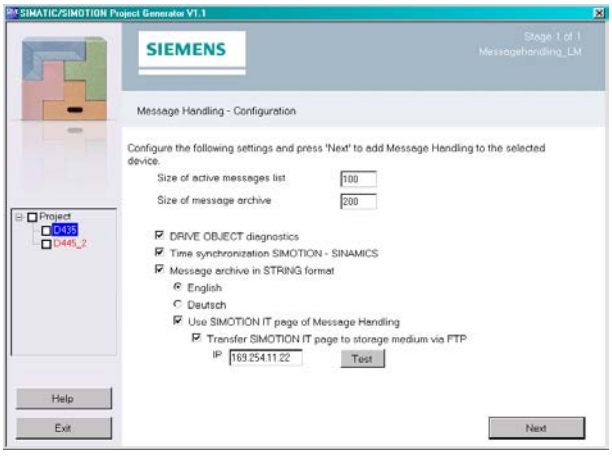


Bild 4-5 Konfiguration Meldungshandling

9. Konfigurieren Sie alle weiteren Geräte nach obigem Muster, indem Sie auf die Schaltfläche **Configure another device** klicken.

   Fertig konfigurierte Geräte werden links außen unterhalb des Projektnamens in grüner Farbe mit Häkchen dargestellt, noch nicht konfigurierte Geräte in roter Farbe, in Arbeit befindliche Geräte orangefarben.

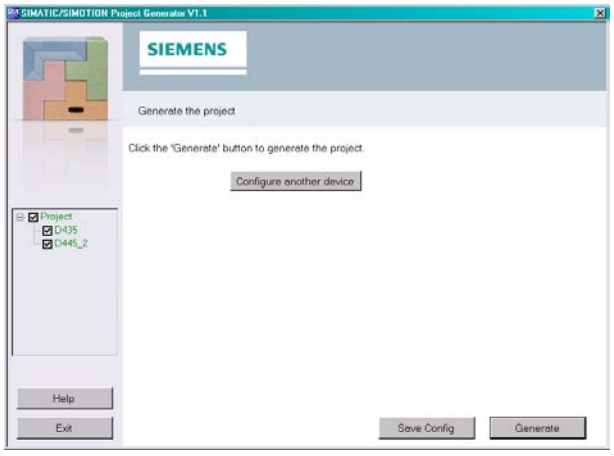


Bild 4-6 Projekt generieren

10. Wenn Sie kein weiteres Gerät mehr konfigurieren (**Configure another device**) und das Projekt abschließen möchten, haben Sie die folgenden Möglichkeiten:

   – Möchten Sie die Konfigurationen der Geräte speichern, aber erst zu einem späteren Zeitpunkt das Projekt generieren, dann klicken Sie auf die Schaltfläche **Save Config** und geben im Explorer den Pfad zum Speicherort ein.

   – Möchten Sie kein weiteres Gerät mehr konfigurieren und das Projekt generieren, dann klicken Sie auf **Generate**.
   Das Projekt wird generiert. Die Dauer richtet sich nach der Art der Konfiguration und wird mit Hilfe eines Fortschrittbalkens angezeigt.
   Ist das Projekt fertig generiert, erscheint im Fenster die Meldung **Generation finished**.

11. Klicken Sie auf die Schaltfläche **Exit**, um den Projektgenerator zu schließen.

Der Projektgenerator ist beendet und liefert folgendes Ergebnis:

Tabelle 4- 1 Ergebnis nach Ausführen des Konfigurations-Skripts

Ergebnis	Bemerkung
Bibliotheken LDPV1 und LMsgHdl wurden ins SIMOTION Projekt importiert	Es erfolgte automatisch eine Versionsprüfung. Stimmen die Versionen nicht überein, wird eine Meldung ausgegeben und der Anwender kann entscheiden, ob die Bibliotheken ausgetauscht werden sollen oder nicht.
ST Units wurden im SIMOTION Projekt angelegt: • fLMsgHdlInit • fLMsgHdl • pLMsgHdl • pGlobalBufferManager	Konfiguration der TOs, Antriebe, Peripheriebaugruppen usw.

Ergebnis	Bemerkung
ST Units wurden in verschiedene Tasks des Ablaufsystems des SIMOTION Projekts eingehängt	**StartupTask**: pLMsgHdlStartupMessagehandling **BackgroundTask**: pLMsgHdlMain, pGlobalBufferManager **TimeFaultTasks**: pLMsgHdlTimeFaultMessage **TimeFaultBackgroundTask**: pLMsgHdlTimeFaultBackgroundMessage **TechnologicalFaultTask**: pLMsgHdlTechnologicalMessage **PeripheralFaultTask**: pLMsgHdlPeripheralMessage **ExecutionFaultTask**: pLMsgHdlExecutionFaultMessage
Kompilierbares Projekt wurde gespeichert	
Daten für die Verwendung der Steuertafel über SIMOTION IT wurden gespeichert	Je nach Auswahl, entweder auf den PC oder auf das Speichermedium des SIMOTION Geräts.

Der Projektgenerator automatisiert verschiedene Handlungen im SIMOTION Projekt und vereinfacht die Inbetriebnahme des Meldungshandlings. Die Anpassungen der Eigenschaften des Tasksystems werden durch den Projektgenerator verändert. Die Bibliotheken können ebenfalls über die SIMOTION SCOUT Funktion **Bibliothek importieren** integriert werden. In diesem Fall muss der Anwender alle notwendigen Handlungen einzeln durchführen, damit das Meldungshandling genutzt werden kann.

Nachträgliches Übertragen der SIMOTION IT Seite auf das Speichermedium des SIMOTION Geräts

Wenn beim Durchlaufen des Projektgenerators die Funktion **Transfer SIMOTION IT web page to storage medium via FTP** nicht angewählt wurde, werden die Daten unter folgendem Pfad im Projektverzeichnis abgelegt: **C:\Dokumente und Einstellungen\<Login-Name>\Local Settings\Temp\LMsgHdl_Files_for_HMI\PGEN_Data_Files\CardFiles**. Um diese Daten zu nutzen, müssen diese auf das Speichermedium des SIMOTION Geräts übertragen werden.

Seit der Version 1.3.0 des Projektgenerators gibt es - neben der bisherigen Möglichkeit, SIMOTION IT Seiten während des Generierens via FTP auf eine Steuerung zu übertragen - auch die Möglichkeit, nur die SIMOTION IT Seiten eines bestehenden Projekts auf eine Steuerung zu übertragen, ohne das Projekt erneut generieren zu müssen. Die detaillierte Beschreibung zum nachträglichen Übertragen der SIMOTION IT Seiten finden Sie im Kapitel *Übertragen von SIMOTION IT Seiten* im Applikationshandbuch *SIMATIC/SIMOTION Projektgenerator*.

4.2.2 Meldungen unterdrücken

Varianz im Meldungshandling durch Setzen von Defines

Durch Setzen von Preprozessor Definitionen (Defines) in den Eigenschaften der Programmunits **pLMsgHdl** und **fLMsgHdl** ist es möglich bestimmte Funktionalitäten im Meldungshandling außerhalb der durch das Skript durchgeführten Varianz auszuschalten.

Die Defines können über rechte Maustaste (Kontextmenü) auf die Unit **pLMsgHdl -> Eigenschaften** im Register **Weitere Einstellungen** eingegeben werden. Da die Unit schreibgeschützt ist, erscheint ein Fenster für die Paßwortabfrage. Klicken Sie auf die Schaltfläche **Abbrechen**. Es öffnet sich das Fenster ST-Quelle Eigenschaften. Detaillierte

Informationen dieser Funktion finden Sie auch in der SIMOTION Online-Hilfe unter Preprozessor -> Eigenschaften einer ST-Quelle ändern.

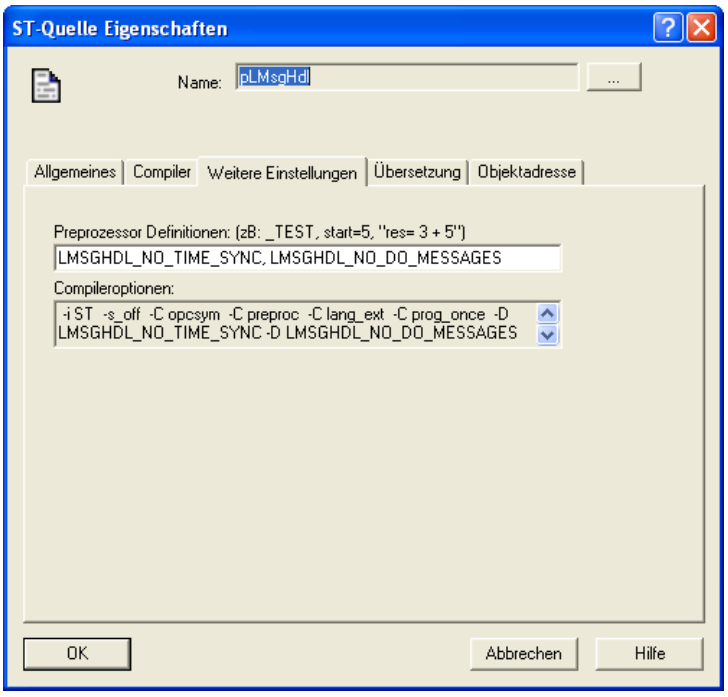


Bild 4-7 Meldungen unterdrücken (Beispiel)

Nachdem Preprozessor Definitionen gesetzt sind, muss die Programmunit neu übersetzt werden.

Es gibt folgende Auswahlmöglichkeiten in der Programmunit **pLMsgHdl**:

- LMSGHDL_NO_TECH_FAULT_MESSAGES
  Es werden keine Meldungen durch Technologieobjekte in der TechnologicalFaultTask durch das Meldungshandling ausgewertet.
- LMSGHDL_NO_PERIPHERAL_MESSAGES
  Es werden keine Meldungen durch Peripheriebaugruppen in der PeripheralFaultTask durch das Meldungshandling ausgewertet.
- LMSGHDL_NO_TIME_FAULT_MESSAGES
  Es werden keine Meldungen durch Zeitüberlauf in der TimeFaultTask und der TimeFaultBackgroundTask durch das Meldungshandling ausgewertet.
- LMSGHDL_NO_EXECUTION_FAULT_MESSAGES
  Es werden keine Meldungen durch Programmfehler in der ExecutionFaultTask durch das Meldungshandling ausgewertet.
- LMSGHDL_NO_DO_MESSAGES
  Es werden keine Meldungen an Antriebsobjekten auf allen SINAMICS-Baugruppen durch das Meldungshandling ausgewertet.
- LMSGHDL_NO_TIME_SYNC
  Die Uhrzeitsynchronisation der SINAMICS-Baugruppen wird nicht durchgeführt.

- LMSGHDL_NO_HMI_FBS
  Das Programm zur Übergabe der Meldungspuffer an HMI / SIMOTION IT wird nicht aufgerufen.
- LMSGHDL_NO_STRING_MESSAGES
  Das Programm zur Übergabe der Meldungspuffer an HMI / SIMOTION IT übergibt die Informationen des Rohdatenpuffers.

Es gibt folgende Auswahlmöglichkeit in der Programmunit **fLMsgHdl**:

- LMSGHDL_NO_RETAIN_BUFFER
  Wechsel des Datenbereichs des MeldungsLogs des SIMOTION Gerätes von RETAIN nach nicht RETAIN.

Es gibt folgende Auswahlmöglichkeit in der Programmunit **dLMsgHdl**:

- LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED
  alternative Meldungsvariante

4.2.3 Anwenderdefinierte Meldungen anlegen

Anwenderdefinierte Meldungen im Format String ins Meldungshandling integrieren

Der Anwender kann aus seiner eigenen Applikation heraus anwenderdefinierte Meldungen absetzen, die in den Puffern des Meldungshandlings eingetragen werden. Diese Meldungen werden wie Meldungen aus dem System behandelt. Um die Meldungen richtig zuordnen zu können, wird der Meldung eine vom Anwender definierte Identifikationsnummer (ID) mitgegeben. Den Meldungen kann auch eine Maschinenfehlerklasse mitgegeben werden, siehe dazu auch Abschnitt Maschinenfehlerklassen definieren (Seite 66).

Die vom Anwender selbst definierten Meldungen werden in die Unit **fLMsgHdlInit** eingetragen. Die Regeln nach denen die Meldungen gebildet werden, werden im Folgenden näher erläutert.

- Alle anwenderdefinierten Meldungen werden in der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** der Unit **fLMsgHdlInit** an die globale Variable *gasLMsgHdlUserDefinedMessages* übergeben.
- Die Anzahl der anwenderdefinierten Meldungen liegt im Bereich von 1 bis 99.999. Die Meldungen beginnen mit dem Subindex 0.
- Aufgrund einer schnelleren Laufzeit beim Erzeugen der Meldungstexte im Format String werden die Meldungstexte bei der Verwendung von Beiwerten aufgeteilt. Hierbei wird die Methode verwendet, dass jeweils ein Text von Beiwert bis Beiwert und anschließend der verwendete Beiwert mit Typinformationen angegeben werden muss.
- Die maximale Länge einer anwenderdefinierten Meldung beträgt 160 Zeichen. Der Anwender muss selber darauf achten, dass diese Gesamtlänge nicht überschritten wird. Geschieht dies dennoch, werden die hinteren Zeichen abgeschnitten.

- Für den Bereich der anwenderdefinierten Meldungen wird ein ARRAY OF Struct verwendet.
- Anwenderdefinierte Meldungen können bis zu vier Beiwerte enthalten.
  Aus diesem Grund müssen die Meldungen, die Beiwerte ans Meldungshandling übertragen, nach folgender Struktur aufgeteilt werden.

Tabelle 4- 2 Struktur für anwenderdefinierte Meldungen im Format String *sLMsgHdlUserMessagesType*

Parameter	Datentyp	Beschreibung
sgLMsgHdlTextPart1	STRING[160]	Erster Teil der Meldung im Format String bis zum Beiwert 1.
ab8LMsgHdlAdditionalValue1	sLMsgHdlAdditionalValueType	Nummer und Format des einzusetzenden Beiwertes.
sgLMsgHdlTextPart2	STRING[90]	Zweiter Teil der Meldung im Format String von Beiwert 1 bis Beiwert 2.
ab8LMsgHdlAdditionalValue2	sLMsgHdlAdditionalValueType	Nummer und Format des einzusetzenden Beiwertes.
sgLMsgHdlTextPart3	STRING[50]	Dritter Teil der Meldung im Format String von Beiwert 2 bis Beiwert 3.
ab8LMsgHdlAdditionalValue3	sLMsgHdlAdditionalValueType	Nummer und Format des einzusetzenden Beiwertes.
sgLMsgHdlTextPart4	STRING[50]	Vierter Teil der Meldung im Format String von Beiwert 3 bis Beiwert 4.
ab8LMsgHdlAdditionalValue4	sLMsgHdlAdditionalValueType	Nummer und Format des einzusetzenden Beiwertes.

Tabelle 4- 3 Struktur für Beiwerte *sLMsgHdlAdditionalValueType*

Parameter	Datentyp	Initialwert	Beschreibung
b8ValueNumber	BYTE	1	Hier können folgende Werte übergeben werden: 0: kein Beiwert an dieser Stelle (b8ValueType ist dann ohne Bedeutung) 1: additionalValue1 2: additionalValue2 3: functionBlockId 4: errorCode 5: additionalValueREAL
b8ValueType	BYTE	0	Hier können folgende Werte übergeben werden: 0: Anzeigeformat dezimal \%d 1: Anzeigeformat hexadezimal \%X 2: Anzeigeformat float \%lf

Beispiel einer anwenderdefinierten Meldung mit vier Beiwerten:

Tabelle 4- 4 Auszug aus der Programmunit **cPublic**

```
//====================================================================
//    constants for definition of user defined message texts
//====================================================================
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_1             : BYTE := 1;
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_2             : BYTE := 2;
```

```
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_FB_ID         : BYTE := 3;
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_ERROR_CODE    : BYTE := 4;
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_REAL          : BYTE := 5;

LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT                : BYTE := 0;
LMSGHDL_USER_MESSAGE_VALUE_TYPE_HEX                 : BYTE := 1;
LMSGHDL_USER_MESSAGE_VALUE_TYPE_REAL                : BYTE := 2;
```

'Anwenderdefinierte Meldung 8. FB/FC:/3/%d. Zusatzinfo1:/1/%d, Zusatzinfo2:/2/%d, ErrorCode:/4/%X'

Diese Meldung muss wie folgt im Meldungshandling eingetragen werden:

Tabelle 4- 5 Beispiel

```
// Anzeige der functionBlock Id
userDefinedMessages[7].sgLMsgHdlTextPart1 := 'Anwenderdefinierte Meldung
8. FB/FC:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue1.b8ValueNumber := 3;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue1.b8ValueType := 0;
// Anzeige additionalValue1
userDefinedMessages[7].sgLMsgHdlTextPart2 := '. Zusatzinfo1:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue2.b8ValueNumber := 1;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue2.b8ValueType := 0;
// Anzeige additionalValue2
userDefinedMessages[7].sgLMsgHdlTextPart3 := ', Zusatzinfo2:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue3.b8ValueNumber := 2;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue3.b8ValueType := 0;
// Anzeige errorCode
userDefinedMessages[7].sgLMsgHdlTextPart4 := ', ErrorCode:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue4.b8ValueNumber := 4;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue4.b8ValueType := 1;
```

Übergibt eine anwenderdefinierte Meldung keine Beiwerte, wird der Meldungstext in das Element *sgLMsgHdlTextPart1* eingetragen. Die anderen Teile der Struktur müssen nicht befüllt werden. Die Summe aller vom Anwender selbst definierten Meldungen muss in die Konstante LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS in die Unit **cPublic** eingetragen werden. Die Anzahl der Meldungen aus den FBs / FCs wird in der Konstanten LMSGHDL_NUMBER_OF_FUNCTION_BLOCK_IDS in der Unit **cPublic** gesetzt. Die Meldungstexte die bei einer Meldung durch FBs / FCs ausgegeben werden sollen, sind Bestandteil der anwenderdefinierten Meldungen und müssen nicht gesondert eingetragen werden. Die Zählung der anwenderdefinierten Meldungen muss immer bei 1 (Eins) beginnen, bis zur Anzahl des jeweiligen Typs.

Soll ein REAL-Wert in einer der bisherigen *additionalValues1* oder *2* ausgegeben werden, kann dies wie folgt gemacht werden:

additionalValue1DINT := DWORD_TO_DINT(REAL_TO_DWORD(2.45286))

4.2.4 Einbettung AlarmS-Verfahren oder Bitmeldeverfahren

AlarmS-Verfahren

AlarmS kann vom Anwender aktiviert werden, indem die Konstante LMSGHDL_ALARM_S_USER_MESSAGES in der Unit **cPublic** der Bibliothek **LMsgHdl** auf TRUE gesetzt wird.

Ist AlarmS aktiviert, wird zu jeder anwenderdefinierten Meldung der zugehörige AlarmS ausgelöst. Dies geschieht innerhalb des Aufrufs der Funktion **FCLMsgHdlWriteUserMessageToBuffer** und **FCLMsgHdlWriteFBFCMessageToBuffer** zur Übergabe von anwenderdefinierten Meldungen.

Nicht jeder anwenderdefinierten Meldung muss auch zwingend eine AlarmS-Meldung zugeordnet werden. Durch Übergabe von STRUCTALARMID#NIL im Array *alarmSInfo* der Programmunit **fMsgHdlInit** der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** wird dem Meldungshandling mitgeteilt, dass hier keine AlarmS-Meldung erzeugt werden soll. Die systemseitige Quittierung der aktiven AlarmS-Meldungen wird bei Quittierung aller anstehenden Meldungen durch das Meldungshandling durchgeführt.

Die Meldungsprojektierung wird in SIMOTION SCOUT durchgeführt. Wenn das Projekt markiert ist, klicken Sie im Kontextmenü (rechte Maustaste) auf **Meldungen -> projektieren**. Es erscheint das Fenster **Meldungsprojektierung**. Nachdem die anwenderdefinierten AlarmS-Meldungen im Projekt editiert, bzw. importiert wurden, müssen diese im Meldungshandling bekanntgemacht werden. Detaillierte Informationen zur Meldungsprojektierung in SIMOTION SCOUT finden Sie in der Online-Hilfe unter den Indexeintrag *Meldungsprojektierung*.

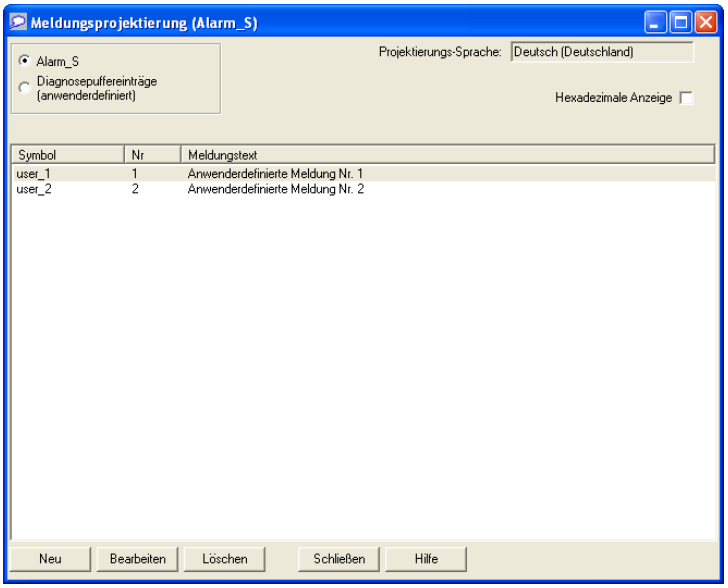


Bild 4-8 AlarmS-Meldungen (Beispiel)

Die Parametrierung der AlarmS-Meldungen wird in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** durch den Anwender durchgeführt. Die beiden in der oberen Abbildung zu sehenden Meldungen werden an die globale Variable *gasLMsgHdlAlarmSInfo* übergeben.

Tabelle 4- 6 Auszug aus der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling**

```
alarmSInfo[0].sAlarmId               := _alarm.user_1;
alarmSInfo[0].boMessageRequiresAck  := TRUE;
alarmSInfo[1].sAlarmId               := _alarm.user_2;
alarmSInfo[1].boMessageRequiresAck  := FALSE;
```

Für jede AlarmS-Meldung ist das Symbol der Meldung und die Art der Meldung vom Anwender zu übergeben. Das Symbol muss mit dem Namensraum für AlarmS _Alarm gefolgt von einem Punkt angegeben werden. Unter der Art der Meldung ist gemeint, ob es sich um einen quittierungspflichtigen Alarm handelt oder nicht. AlarmSQ ist quittierungspflichtig durch das HMI, AlarmS nur durch das System.

Hinweis

Bei der Quittierung von AlarmSQ-Meldungen über das Meldungshandling, werden nur dort die AlarmSQ-Meldungen quittiert. Die Anzeige dieser Meldungen im HMI muss separat vom Anwender quittiert werden.

Die bei aktivem AlarmS-Verfahren ausgelöste Meldung wird dabei durch die Eventnummer (Nummer der anwenderdefinierten Meldung) bestimmt, die den Funktionen **FCLMsgHdlWriteUserMessageToBuffer** und **FCLMsgHdlWriteFBFCMessageToBuffer** übergeben wird. Hierbei entspricht die Eventnummer dem Subindex -1 in *gasLMsgHdlAlarmSInfo*. Somit wird der AlarmS wie folgt ausgelöst:

```
gasLMsgHdlAlarmSInfo[0].sAlarmId    := _alarm.user_1;
```

Somit kann zu einer bestimmten anwenderdefinierten Meldung der entsprechende AlarmS zugeordnet werden, ohne dass beide Meldungen die gleiche Nummer im Meldungshandling benötigen. Soll eine anwenderdefinierte Meldung keinen AlarmS auslösen, wird die entsprechende *sAlarmId* nicht mit einem Wert belegt (STRUCTALARMID#NIL). Weiterhin ist zu beachten, dass jede AlarmS-Meldung nur ein einziges Mal aktiv sein kann. Eine aktive AlarmS-Meldung kann also erst nach deren Quittierung erneut ausgelöst werden. Die systemseitige Quittierung der aktiven AlarmS-Meldungen wird bei der Quittierung aller anstehenden Meldungen durch das Meldungshandling mit durchgeführt. Weitere Informationen zu AlarmS finden Sie in der SIMOTION SCOUT Online-Hilfe.

Bitmeldeverfahren

Das Bitmeldeverfahrens kann aktiviert werden, indem die Konstante LMSGHDL_MESSAGE_BIT_USER_MESSAGES in der Unit **cPublic** der Bibliothek **LMsgHdl** auf TRUE gesetzt wird. Die aktiven Meldungen werden dem Bitmeldeverfahren im Array *gab16LMsgHdlEventFlag* aus der Programmunit **fLMsgHdl** übergeben. Wird eine anwenderdefinierte Meldung durch den Aufruf der Funktion **FCLMsgHdlWriteUserMessageToBuffer** und **FCLMsgHdlWriteFBFCMessageToBuffer** gesetzt, so wird das entsprechende Bit in *gab16LMsgHdlEventFlag* ebenfalls gesetzt. Das gesetzte Bit entspricht dabei der Meldungsnummer -1 der anwenderdefinierten Meldung. Hierbei werden die Meldungsbits in einem Array vom Typ WORD gesetzt. Daraus ergibt sich das aus der Meldungsnummer zu setzende Bit wie folgt:

```
gab16LMsgHdlEventFlag[i32EventNumber-1/16].(i32EventNumber-1 MOD 16) :=
TRUE
```

In der Variablen *gab16LMsgHdlEventFlag* wird das zur ausgelösten Meldung entsprechende Bit gesetzt. Es wird angezeigt, bzw. übergeben, ob die entsprechende Bitmeldung quittiert wurde, bzw. quittiert werden muss.

Bei der Erstellung der Meldungen ist darauf zu achten, dass der Index der Meldung immer mit 1 (Eins) beginnt. Eine 0 (Null) ist nicht zulässig.

4.2.5 Maschinenfehlerklassen definieren

Vordefinierte Maschinenfehlerklassen

Hinweis

Alle Eingaben oder Änderungen, die vom Anwender erstellt wurden, werden vom Konfigurations-Skript gesichert, wenn dieses nochmals vom Anwender gestartet werden muss.

In der Programmunit **fLMsgHdlInit** sind bereits folgende mögliche Maschinenfehlerklassen vordefiniert.

Tabelle 4- 7 Vordefinierte Maschinenfehlerklassen in der Unit **cPublic**

```
//=============================================================
//         Defines for machine error classes
//=============================================================
LMSGHDL_NO_MACHINE_ERROR_CLASS : SINT :=-1;
LMSGHDL_MACHINE_ERROR_CLASS0 : SINT   :=0;
LMSGHDL_MACHINE_ERROR_CLASS1 : SINT   :=1;
LMSGHDL_MACHINE_ERROR_CLASS2 : SINT   :=2;
LMSGHDL_MACHINE_ERROR_CLASS3 : SINT   :=3;
LMSGHDL_MACHINE_ERROR_CLASS4 : SINT   :=4;
```

Es gibt drei unterschiedliche Quellen von denen aus eine Maschinenfehlerklasse gesetzt werden kann:

- Maschinenfehlerklasse durch anwenderdefinierte Meldung
- Maschinenfehlerklasse durch Peripheriemeldungen
- Maschinenfehlerklasse durch anwenderdefinierte FBs / FCs

Diese werden nachfolgend beschrieben.

Maschinenfehlerklasse durch anwenderdefinierte Meldung

Beim Auslösen von anwenderdefinierten Meldungen wird jeder Meldung eine Eventnummer zugewiesen. In eine Tabelle wird die zu einer Eventnummer entsprechende Maschinenfehlerklasse eingegeben.

Tabelle 4- 8 Zuordnung der Eventnummer zur Maschinenfehlerklasse - Beispiel

Eventnummer	**Maschinenfehlerklasse**
1	Maschinenfehlerklasse 4
2	Maschinenfehlerklasse 3
3	Maschinenfehlerklasse 1
4	Maschinenfehlerklasse 1
5	Maschinenfehlerklasse 2
...	...

Diese Tabelle wird über das globale Array *gai8LMsgHdlUserDefinedMachineErrors* im Meldungshandling bereitgestellt und muss vom Anwender selbst angepasst werden. Die Initialisierung der anwenderdefinierten Meldung kann in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** gemacht werden. Hierbei entspricht der Subindex in *userDefinedMachineErrors* der Eventnummer -1 der entsprechenden anwenderdefinierten Meldung.

```
userDefinedMachineErrors[0] := LMSGHDL_MACHINE_ERROR_CLASS4;
userDefinedMachineErrors[1] := LMSGHDL_MACHINE_ERROR_CLASS3;
userDefinedMachineErrors[2] := LMSGHDL_MACHINE_ERROR_CLASS1;
```

Maschinenfehlerklasse durch Peripheriemeldungen

Peripheriemeldungen können ebenfalls einer Maschinenfehlerklasse zugewiesen werden. Hierbei wird jedem Peripheriegerät eine Maschinenfehlerklasse zugewiesen. Die Maschinenfehlerklasse wird nur gesetzt, wenn eine negative Meldung, wie z. B. **Stationsausfall** ansteht, nicht aber beispielsweise bei einer **Stationswiederkehr**.

Die Maschinenfehlerklassen für Peripheriemeldungen werden in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** gesetzt. Die Übergabe der Maschinenfehlerklasse für den Ausfall einer Peripheriemeldung geschieht in der globalen Variable *gasLMsgHdlPeripheralDevices*. Hier wird die Meldungsklasse in der Variablen *peripheralDevices[0].i8MachineErrorClass* gesetzt. Der zur Peripheriebaugruppe gehörende Subindex zum Setzen der Maschinenfehlerklasse kann in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlInitProjectInfo** nachgesehen werden. Dort werden die Informationen zu Peripheriebaugruppen durch das Skript automatisch gesetzt.

Tabelle 4- 9 Zuordnung der Peripherie zur Maschinenfehlerklasse - Beispiel

Peripherie	Maschinenfehlerklasse
Peripheriegerät 0	Maschinenfehlerklasse 4
Peripheriegerät 1	Maschinenfehlerklasse 3
Peripheriegerät 2	Maschinenfehlerklasse 2

```
peripheralDevices[0].i8MachineErrorClass := LMSGHDL_MACHINE_ERROR_CLASS4;
peripheralDevices[1].i8MachineErrorClass := LMSGHDL_MACHINE_ERROR_CLASS3;
peripheralDevices[2].i8MachineErrorClass := LMSGHDL_MACHINE_ERROR_CLASS2;
```

Maschinenfehlerklasse durch anwenderdefinierte FBs / FCs

Da in einem vom Anwender definierten Funktionsbaustein, bzw. einer Funktion Fehler mit unterschiedlicher Schwere auftreten können, gibt es hier die Möglichkeit in Abhängigkeit einer Fehlerklasse eine entsprechende Maschinenfehlerklasse zu setzen. Aus den Funktionsbausteinfehlerklassen wird mit Hilfe einer Matrix die Maschinenfehlerklasse generiert.

Tabelle 4- 10 Zuordnung der FBs / FCs zu den Maschinenfehlerklassen (Maschinen-FK)

ID FB /FC	Fehlerklasse 0	Fehlerklasse 1	Fehlerklasse 2	Fehlerklasse 3
ID FB / FC 1	Maschinen-FK 1	Maschinen-FK 1	Maschinen-FK 2	Maschinen-FK 2
ID FB / FC 2	Maschinen-FK 1	Maschinen-FK 2	Maschinen-FK 3	Maschinen-FK 4
ID FB / FC 3	Maschinen-FK 1	Maschinen-FK 2	Maschinen-FK 3	Maschinen-FK 4
ID FB / FC 4	Maschinen-FK 1	Maschinen-FK 2	Maschinen-FK 3	Maschinen-FK 4
ID FB / FC 5	Maschinen-FK 1	Maschinen-FK 1	Maschinen-FK 2	Maschinen-FK 3

Jeder Funktionsbaustein / jede Funktion kann hierbei bis zu vier unterschiedliche Maschinenfehlerklassen 0 bis 3 auslösen. Mit Hilfe dieser Matrix kann parametriert werden, wie relevant ein Fehler für den Gesamtbetrieb der Maschine ist. Im obigen Beispiel ist zu erkennen, dass Fehler an den FBs / FCs 1 und 5 weniger hohe Maschinenfehlerklassen auslösen, als Fehler an den anderen FBs / FCs.

Die Projektierung dieser Matrix wird hierbei in der Programmunit **fLMsgHdlInit** in der Funktion **FCLMsgHdlUserDefinedInfoForMessageHandling** durchgeführt. Die Matrix wird in der globalen Variable *gasLMsgHdlFBFCMachineErrorClasses* übergeben.

```
fBFCMachineErrorClasses[0].ai8ErrorClass[0] :=
LMSGHDL_MACHINE_ERROR_CLASS1;
```

```
fBFCMachineErrorClasses[0].ai8ErrorClass[1] :=
LMSGHDL_MACHINE_ERROR_CLASS1;
fBFCMachineErrorClasses[0].ai8ErrorClass[2] :=
LMSGHDL_MACHINE_ERROR_CLASS2;
fBFCMachineErrorClasses[0].ai8ErrorClass[3] :=
LMSGHDL_MACHINE_ERROR_CLASS2;
```

Die aktuell anstehenden Maschinenfehlerklassen werden durch das Meldungshandling an zwei Variablen übergeben. Diese Variablen sind in der Programmunit **fLMsgHdl** deklariert.

- *gi8LMsgHdlMachineErrorClass*: Hier wird die aktuell höchste Maschinenfehlerklasse angezeigt, die in der Applikation aktiv ist.
- *gb32LMsgHdlMachineErrorClasses*: Hier werden bitcodiert alle aktuell aktiven Maschinenfehlerklassen angezeigt. Da diese Variable vom Typ DWORD ist kann es maximal 32 unterschiedliche Maschinenfehlerklassen im Meldungshandling geben (0 bis 31). Jede dieser Maschinenfehlerklassen wird über das entsprechende Bit in der Variablen angezeigt.

4.3 Meldungen über SIMOTION IT anzeigen lassen

Funktionalität

Über die SIMOTION IT Seite können die anstehenden Meldungen angezeigt werden.

Systemvoraussetzungen

- SIMOTION Firmware V4.1 SP4 oder höher
- Internet Explorer 6, bzw. Mozilla Firefox 3.5 oder höher

Anzeigen von Meldungen über SIMOTION IT

1. Starten Sie sie Verbindung zum SIMOTION Gerät, geben Sie dazu die IP-Adresse in die Adress-Zeile des Browsers ein.
2. Öffnen Sie die Seite **User's Area**.
3. Öffnen Sie die Seite **Messages**.

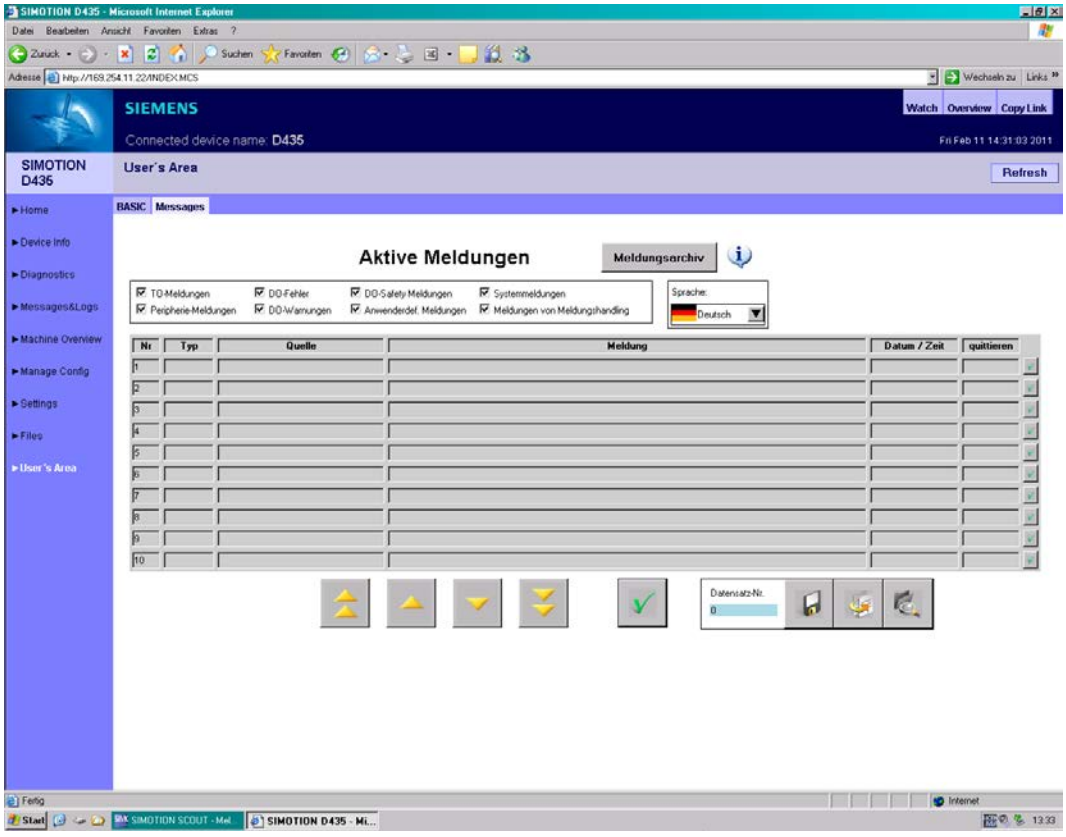


Bild 4-9 Einstiegsseite Steuertafel Meldungshandling

4.4 Wichtige, häufig vom Anwender genutzte Variablen

Auflistung der vom Anwender häufig genutzten Variablen

Das Meldungshandling ist eine komplexe Applikation mit vielen Variablen. Da nicht alle Variablen vom Anwender immer genutzt werden, ist in der folgenden Tabelle eine Übersicht der wichtigsten Variablen zusammengestellt.

Tabelle 4- 11 Wichtige, häufig genutzte Variablen im Meldungshandling

Unit	Variable	Bedeutung
fLMsgHdl	gb32LMsgHdlMachineErrorClasses	Hier werden bitcodiert alle aktuell aktiven Maschinenfehlerklassen angezeigt. Die maximale Anzahl ist 32 (0 bis 31).
	gi8LMsgHdlMachineErrorClass	Anzeige der höchstprior, aktiven Maschinen-Fehlerklasse.
	gab16LMsgHdlEventFlag	Bits für Bitmeldeverfahren Bitarray zur Anzeige der aktiven anwenderdefinierten Meldungen für das Bitmeldeverfahren.
	gsLMsgHdlActiveMessageTypes	Bitarray für die Meldungsquellen von aktiven Meldungen.
	gsLMsgHdlMessageLogString	Meldungshistorie im Format String
pLMsgHdl	gboLMsgHdlInitDriveReady	Zeigt an, dass die Initialisierungs-Software des Meldungshandlings in der BackgroundTask durchgelaufen ist und dass das Meldungshandling aktiv ist.
	gboLMsgHdlActivateNewMoMaData	Mit TRUE werden die zur Laufzeit übergebenen Informationen für modulare Maschine aktiviert. Nach Aktivierung wird das Flag vom Meldungshandling wieder zurückgenommen.
	gboLMsgHdlGlobalAcknowledge	Globale Quittierung aller anstehenden Fehler, mit steigender Flanke.
	gi32LMsgHdlNumberOfMessageInLog	Übergabe der Nummer der zu quittierenden Meldung (Nur bei Einzelquittierung.)
	gboLMsgHdlStartChangeLanguage	Mit TRUE, Start der Sprachumschaltung. Wird vom Meldungshandling selbstständig zurückgenommen.
	gu8LMsgHdlActiveLanguage	Übergabe der einzuwechselnden Sprache
	gboLMsgHdlStartWriteComplete MessageLogToStorageMedium	Mit TRUE, Start der Speicherung des aktuellen MeldungsLog auf das Speichermedium des SIMOTION Geräts. Wird vom Meldungshandling selbstständig zurückgenommen.
	gu32LMsgHdlDatasetNoForExport MessageLog	Name der Datei in der der aktuelle MeldungsLog gespeichert werden soll
	gboLMsgHdlUpdateHMI	Aktualisieren der angezeigten aktiven Meldungen.

Unit	Variable	Bedeutung
	gboLMsgHdlUpdateHMILog	Aktualisieren der angezeigten aktiven Meldungshistorie.
	gboLMsgHdlScrollDown	Scroll Down in der Liste der aktiven Meldungen mit steigender Flanke
	gboLMsgHdlScrollDown1	Scroll Down in der Liste der aktiven Meldungen mit steigender Flanke um 1 Zeile
	gboLMsgHdlScrollUp	Scroll Up
	gboLMsgHdlScrollUp1	Scroll Up Down in der Liste der aktiven Meldungen mit steigender Flanke um 1 Zeile
	gboLMsgHdlGoToTop	Springe an den Anfang der aktiven Meldungen.
	gboLMsgHdlGoToEnd	Springe ans Ende der aktiven Meldungen.
	gboLMsgHdlScrollDownLog	Scroll Down in der Meldungshistorie
	gboLMsgHdlScrollDown1Log	Scroll Down um 1 Zeile
	gboLMsgHdlScrollUpLog	Scroll Up
	gboLMsgHdlScrollUp1Log	Scroll Up um 1 Zeile
	gboLMsgHdlGoToTopLog	Springe an den Anfang der Meldungshistorie.
	gboLMsgHdlGoToEndLog	Springe ans Ende der Meldungenhistorie.
	gu8LMsgHdlScrollStep	Anzahl Zeilen für Scroll Down und Scroll Up
	gu8LMsgHdlNumberOfLinesForHMI	Anzahl der Meldungen, die am HMI angezeigt werden können.
	gsLMsgHdlActiveMessageString	Liste der aktiven Meldungen im Format String.
	gsLMsgHdlActiveMsgToHMI	Liste der aktiven Meldungen, die an das HMI ausgegeben werden soll.
	gsLMsgHdlLogMsgToHMI	Liste der Meldehistorie, die an das HMI ausgegeben werden soll.
fLMsgHdlInit	gasLMsgHdlUserDefinedMessages	Beschreibung der anwenderdefinierten Meldungen.

Funktionsbeschreibung 5

5.1 Allgemeines zur Funktionsbeschreibung

Im Meldungshandling sind folgende für den Anwender relevanten FBs und FCs integriert:

- Funktionsbaustein **FBLMsgHdlActiveMsgSgToHMI** (Unit fMsgHdl in LMsgHdl)
- Funktionsbaustein **FBLMsgHdlMsgLogSgToHMI** (Unit fMsgHdl in LMsgHdl)
- Funktionsbaustein **FBLMsgHdlActiveMsgBaseDataToHMI** (Unit fMsgHdl in LMsgHdl)
- Funktionsbaustein **FBLMsgHdlMsgLogBaseDataToHMI** (Unit fMsgHdl in LMsgHdl)
- Funktion **FCLMsgHdlWriteUserMessageToBuffer** (Unit fLMsgHdl)
- Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** (Unit fLMsgHdl)

Diese werden in den folgenden Abschnitten beschreiben.

5.2 Funktionsbaustein FBLMsgHdlActiveMsgSgToHMI

5.2.1 Allgemeines zum Funktionsbaustein

Hinweis

Es können nur Strings der Länge 80 in WinCC flexible verarbeitet werden. Aus diesem Grund ist es notwendig, dass die Meldungstexte aus den aktiven Meldungen in zwei Teil-Strings aufgeteilt werden. Die Länge der Daten kann jedoch bei der Verwendung eines anderen HMI über die Konstante LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI in **cPublic** der Bibliothek **LMsgHdl** geändert werden.

Zur Anzeige der Meldungstexte des globalen Puffers für aktive Meldungen im Format String an ein HMI kommt der Funktionsbaustein **FBLMsgHdlActiveMsgSgToHMI** zum Einsatz.

Der Funktionsbaustein beendet seine Bearbeitung in einem Bearbeitungszyklus der Task, in der er aufgerufen wird. Der FB sollte bevorzugt in der BackgroundTask aufgerufen werden. Er reagiert nur auf fallende Flanken an den entsprechenden Eingängen. Werden bei einem Aufruf mehrere Eingänge gesetzt wird nur die erste Funktion ausgeführt. Die Logikreihenfolge beim Auswerten der Eingangssignale ist dabei:

- updateHMI
- scrollDown1
- scrollDown
- scrollUp1

- scrollUp
- goToTop
- goToEnd

Da immer nur ein Teilabschnitt des gesamten Puffers für aktive Meldungen am HMI angezeigt werden kann, kontrolliert der FB beim Scrollen nach **Oben** bzw. nach **Unten** jeweils selbstständig, ob das entsprechende Ende des Puffers für alle aktiven Meldungen erreicht ist oder nicht. Ist ein Ende erreicht, hat ein neues Verschieben in die entsprechende Richtung keine Aktion im Funktionsbaustein zur Folge.

Über *goToTop* bzw. *goToEnd* kann der Anwender jeweils direkt an das Ende bzw. den Anfang der Liste springen.

Wird z. B. ein Scrollen um 10 Meldungen nach Oben gefordert, aber es sind nur noch 3 Meldungen bis zum oberen Ende des gesamten Puffers, so wird die Liste für HMI auch nur um 3 Meldungen nach oben verschoben.

Über die Konstante LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI der Unit **cPublic** wird festgelegt, wie viele Meldungen maximal an ein HMI übergeben werden können. Über den Eingang *numberOfLinesForHMI* wird dem FB übergeben, wie viele Meldungen er am Ausgang in der Struktur *activeMsgToHMI* tatsächlich bereitstellen soll. Somit ist es möglich über einzelne Instanzen des FB unterschiedliche HMI mit unterschiedlicher Anzahl von Meldungstexten zu versorgen. Die maximale Anzahl der Zeilen in der Konstanten LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI darf nicht überschritten werden. Die Anzahl der zu scrollenden Zeilen, die bei der Betätigung der Eingänge *scrollUp* und *scrollDown* verwendet wird, wird über die Eingangsvariable *numberOfLinesToScroll* festgelegt.

Am Ausgang *activeMsgToHMI* werden die aktiven Meldungen im Format String auf einem HMI ausgegeben.

5.2.2 Schematische Darstellung KOP/FUP

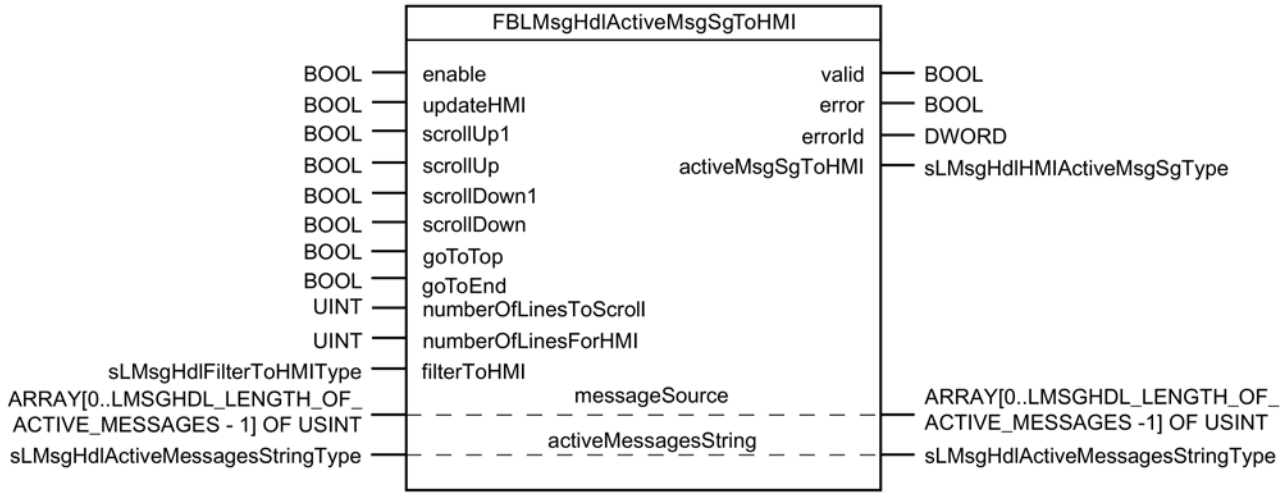


Bild 5-1 Schematische Darstellung KOP/FUP

5.2.3 Eingangs- und Ausgangsparameter des Funktionsbausteins

Der Funktionsbaustein **FBLMsgHdlActiveMsgSgToHMI** hat folgende Eingangs- und Ausgangsparameter:

Tabelle 5- 1 Eingangs- und Ausgangsparameter

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
enable	IN	BOOL	M	FALSE	Freigabe des FB
updateHMI	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der aktuell ausgegebene Datenbereich für das HMI aktualisiert.
scrollUp1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach oben verschoben.
scrollUp	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in der Variablen *numberOfLinesToScroll* übergebenen Wert nach oben verschoben.
scrollDown1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach unten verschoben.
scrollDown	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in der Variablen *numberOfLinesToScroll* übergebenen Wert nach unten verschoben.
goToTop	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der obersten Zeile auf den aktuellsten Eintrag verschoben.
goToEnd	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der untersten Zeile auf den ältesten Eintrag verschoben.
numberOf LinesToScroll	IN	UINT	O	1	Wert der vorgibt, um wie viele Zeilen bei Betätigung von *scrollUp,* bzw. *scrollDown* der Anzeigebereich verschoben werden soll.
numberOf LinesForHMI	IN	UINT	O	1	Gibt die Anzahl an Zeilen (Meldungen) an, die für das HMI ausgegeben werden.
filterToHMI	IN	sLMsgHdl FilterToHMIType	M		Bitstruktur zur Übergabe von Filterinformationen zur Ausgabe an HMI
messageSource	IN/OUT	ARRAY [0..LMSGHDL _LENGTH_OF _ACTIVE_ MESSAGES - 1] OF USINT	M		Hier wird dem FB für die Filterung der Meldungen die jeweilige Meldungsquelle übergeben. Diese wird selbstständig vom Meldungshandling in **pLMsgHdl** *gau8LMsgHdlActMessageStringMessageSource* gebildet.
Active MessagesString	IN/OUT	sLMsgHdl Active Messages StringType	M	-	Übergabe des aktuellen Meldepuffers im Format String.
valid	OUT	BOOL	-	FALSE	Anzeige der Gültigkeit der Werte an den Ausgängen

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
error	OUT	BOOL	-	FALSE	Zeigt an, ob ein Fehler bei der Bearbeitung des FB aufgetreten ist.
errorId	OUT	DWORD	-	16#00000000	Gibt die Nummer des Fehlers zurück, der aufgetreten ist.
activeMsgToHMI	OUT	sLMsgHdl HMIActive MsgSgType	-	-	Rückgabe der aktiven Meldungen als String für die Anzeige auf einem HMI.

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

Tabelle 5- 2 Fehlermeldungen

Fehlernummer [HEX]	Beschreibung
0	fehlerfrei
9997	Die Anzahl der im HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Anzahl der Zeilen für das HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in der Unit **cPublic**)
9998	Die Anzahl der am HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Länge des übergebenen Puffers (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES oder LMSGHDL_LENGTH_OF_MESSAGE_LOG in der Unit **cPublic**)

5.2.4 Struktur für Parameterübergabe

Die Struktur *sLMsgHdlHMIActiveMsgType* ist wie folgt aufgebaut.

Tabelle 5- 3 Struktur für aktive Meldungen als String an ein HMI

Parameter	Datentyp	Initialwert	Beschreibung
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array für Information, welche Nummer die entsprechende Meldung im Meldungspuffer für aktive Meldungen hat
asgMessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[11]	Leerstring	Array für Information über Stufe der Meldung (Störung, Warnung, Fehler, Information)
asgMessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[64]	Leerstring	Array für Information über Quelle der Meldung
asgMessageText1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Leerstring	Array für sprachabhängigen Meldungstext Abschnitt 1 (eine Meldung darf maximal 160 Zeichen lang sein)

Parameter	Datentyp	Initialwert	Beschreibung
asgMessageText2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Leerstring	Array für sprachabhängigen Meldungstext Abschnitt 2 (eine Meldung darf maximal 160 Zeichen lang sein)
asgMessageOccured	ARRAY[0..LMSGHDL_NUMBER_OF_ LINES_MAX_FOR_HMI - 1] OF STRING[23]	Leerstring	Zeitstempel, wann die entsprechende Meldung aufgetreten ist.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[17]	Leerstring	Array für Information über die Quittierungsart der Meldung

5.3 Funktionsbaustein FBLMsgHdlMsgLogSgToHMI

5.3.1 Allgemeines zum Funktionsbaustein

Hinweis

Es können nur Strings der Länge 80 in WinCC flexible verarbeitet werden. Aus diesem Grund ist es notwendig, dass die Meldungstexte aus den aktiven Meldungen in zwei Teil-Strings aufgeteilt werden. Die Länge der Daten kann jedoch bei der Verwendung eines anderen HMIs über die Konstante LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI in **cPublic** der Bibliothek **LMsgHdl** geändert werden.

Zur Anzeige der Meldungstexte des globalen Puffers für MeldungsLog im Format String an ein HMI kommt der Funktionsbaustein **FBLMsgHdlMsgLogSgToHMI** zum Einsatz.

Verhalten wie im Funktionsbaustein FBLMsgHdlActiveMsgSgToHMI (Seite 73) beschrieben.

5.3.2 Schematische Darstellung KOP/FUP

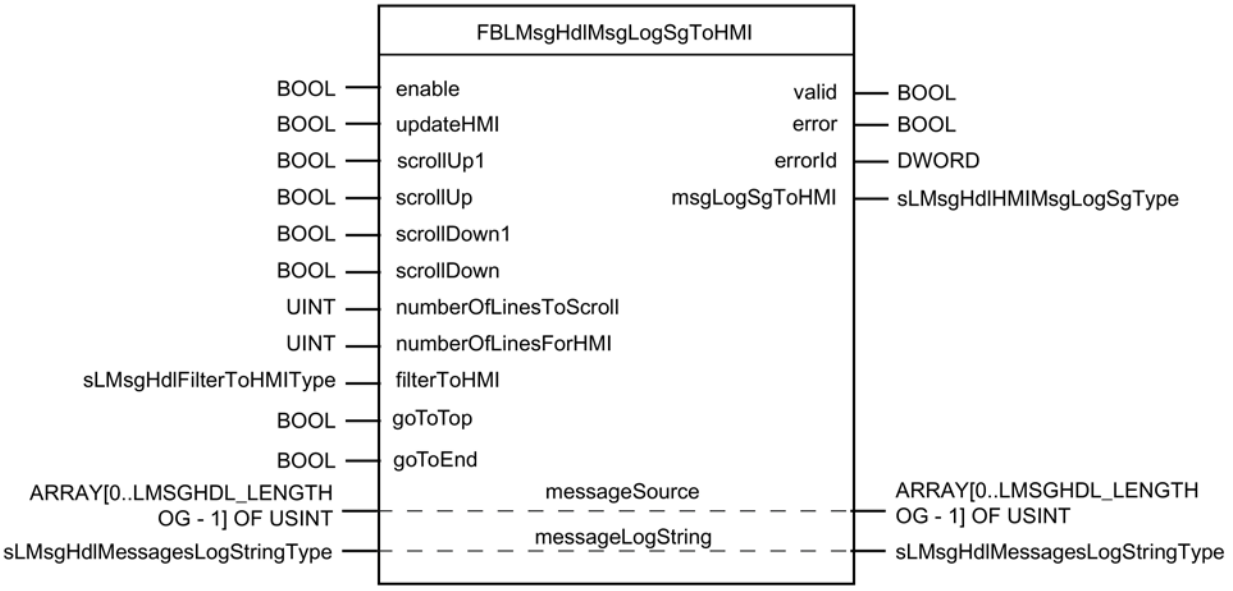


Bild 5-2 Schematische Darstellung KOP/FUP

5.3.3 Eingangs- und Ausgangsparameter des Funktionsbausteins

Der Funktionsbaustein **FBLMsgHdlMsgLogSgToHMI** hat folgende Eingangs- und Ausgangsparameter:

Tabelle 5- 4 Eingangs- und Ausgangsparameter

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
enable	IN	BOOL	M	FALSE	Freigabe des FB
updateHMI	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der aktuell ausgegebene Datenbereich für das HMI aktualisiert.
scrollUp1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach oben verschoben.
scrollUp	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in *numberOfLinesToScroll* übergebenen Wert nach oben verschoben
scrollDown1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach unten verschoben
scrollDown	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in *numberOfLinesToScroll* übergebenen Wert nach unten verschoben.

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
goToTop	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der obersten Zeile auf den aktuellsten Eintrag verschoben.
goToEnd	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der untersten Zeile auf den ältesten Eintrag verschoben.
numberOfLines ToScroll	IN	UINT	O	1	Wert der vorgibt, um wie viele Zeilen bei Betätigung von *scrollUp,* bzw. *scrollDown* der Anzeigebereich verschoben wird.
numberOfLines ForHMI	IN	UINT	O	1	Gibt die Anzahl an Zeilen (Meldungen) an, die für das HMI ausgegeben werden.
filterToHMI	IN	sLMsgHdl FilterToHMIType	O		Hier werden die Informationen übergeben, welche Meldungsquellen am Ausgang des FB angezeigt bzw. nicht angezeigt werden sollen. Ist im Meldungshandling die Verwendung von SIMOTION IT angewählt, werden diese Informationen in *gsLMsgHdlFilterToHMI* der Unit pLMsgHdl bereitgestellt und sollten als VAR_IN_OUT an den FB übergeben werden.
messageSource	IN/OUT	ARRAY [0..LMSGHDL_ LENGTH_OF -1] OF UINT	M		Hier müssen dem FB die zu den Meldungen gehörenden Meldungsquellen im Format USINT übergeben werden. Die Meldungsquellen im Format USINT werden vom Meldungshandling erzeugt und liegen in *gau8LMsgHdlMessageLogStringMessageSource* der Unit pLMsgHdl. Diese Variable muss hier als VAR_IN_OUT übergeben werden.
messageLogString	IN/OUT	sLMsgHdl Message LogStringType	M	-	Übergabe des MeldeLog-Puffers im Format String.
vaild	OUT	BOOL	-	FALSE	Anzeige der Gültigkeit der Werte an den Ausgängen
error	OUT	BOOL	-	FALSE	Zeigt an, ob ein Fehler bei der Bearbeitung des FB aufgetreten ist.
errorId	OUT	DWORD	-	16#00000000	Gibt die Nummer des Fehlers zurück, der aufgetreten ist.
msgLogToHMI	OUT	sLMsgHdlHMI MsgLogSgType	-	-	Rückgabe der Meldungen aus dem MeldungsLog im Format String für die Anzeige auf HMI.

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter, IN/OUT = Durchgangsparameter

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

Tabelle 5- 5 Fehlermeldungen

Fehlernummer [HEX]	Beschreibung
0	fehlerfrei
9997	Die Anzahl der im HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Anzahl der Zeilen für das HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in der Unit **cPublic**)
9998	Die Anzahl der am HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Länge des übergebenen Puffers (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES oder LMSGHDL_LENGTH_OF_MESSAGE_LOG in der Unit **cPublic**)

5.3.4 Struktur für Parameterübergabe

Die Struktur *sLMsgHdlHMIMsgLogSgType* ist wie folgt aufgebaut.

Tabelle 5- 6 Struktur für Meldungen im MeldungsLog als String an ein HMI

Parameter	Datentyp	Initialwert	Beschreibung
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array für Information, welche Nummer die entsprechende Meldung im Meldungspuffer für den MeldungsLog hat.
asgMessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[11]	Leerstring	Array für Information über Stufe der Meldung (Störung, Warnung, Fehler, Information).
asgMessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[64]	Leerstring	Array für Information über Quelle der Meldung.
asgMessageText1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Leerstring	Array für sprachabhängigen Meldungstext Abschnitt 1 (eine Meldung darf maximal 160 Zeichen lang sein).
asgMessageText2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Leerstring	Array für sprachabhängigen Meldungstext Abschnitt 2 (eine Meldung darf maximal 160 Zeichen lang sein).
asgMessageOccured	ARRAY[0..LMSGHDL_NUMBER_OF_ LINES_MAX_FOR_HMI - 1] OF STRING[23]	Leerstring	Zeitstempel, wann die entsprechende Meldung aufgetreten ist.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[17]	Leerstring	Array für Information über die Quittierungsart der Meldung.
asgMessageGone	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[23]	Leerstring	Zeitstempel, wann die entsprechende Meldung gegangen ist.

Übersicht der Anwenderkonstanten und Strukturen, siehe Abschnitt Öffentliche Konstanten (Seite 22).

5.4 Funktionsbaustein FBLMsgHdlActiveMsgBaseDataToHMI

5.4.1 Allgemeines zum Funktionsbaustein

Zur Anzeige der Meldungen des globalen Puffers für aktive Meldungen im Format Rohdaten an ein HMI kommt der Funktionsbaustein **FBLMsgHdlActiveMsgBaseDataToHMI** zum Einsatz.

Verhalten wie im Funktionsbaustein FBLMsgHdlActiveMsgSgToHMI (Seite 73) beschrieben.

Siehe auch

Interpretation der Rohdaten (Seite 121)

5.4.2 Schematische Darstellung KOP/FUP

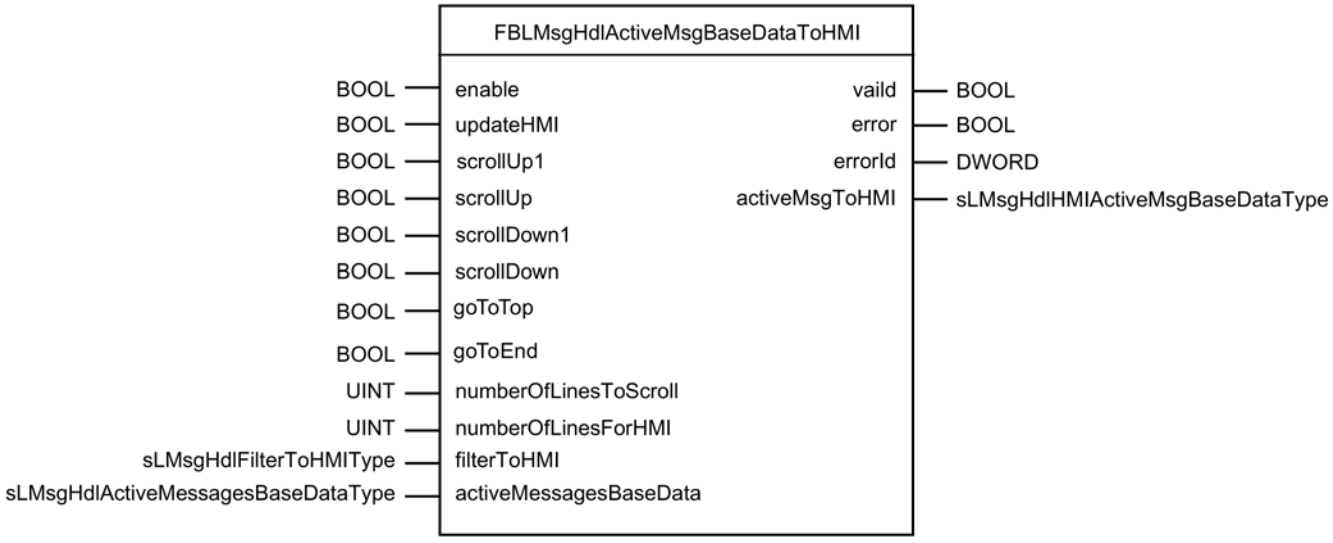


Bild 5-3 Schematische Darstellung KOP/FUP

5.4.3 Eingangs- und Ausgangsparameter des Bausteins

Der Funktionsbaustein **FBLMsgHdlActiveMsgBaseDataToHMI** hat folgende Eingangs- und Ausgangsparameter:

Tabelle 5- 7 Eingangs- und Ausgangsparameter

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
enable	IN	BOOL	M	FALSE	Freigabe des FB
updateHMI	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der aktuell ausgegebene Datenbereich für das HMI aktualisiert.
scrollUp1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach oben verschoben.
scrollUp	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in der Variablen *numberOfLinesToScroll* übergebenen Wert nach oben verschoben.
scrollDown1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach unten verschoben.
scrollDown	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in der Variablen *numberOfLinesToScroll* übergebenen Wert nach unten verschoben.
goToTop	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der obersten Zeile auf den aktuellsten Eintrag verschoben.
goToEnd	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der untersten Zeile auf den ältesten Eintrag verschoben.
numberOfLines ToScroll	IN	UINT	O	1	Wert der vorgibt, um wie viele Zeilen bei Betätigung von *scrollUp*, bzw. *scrollDown* der Anzeigebereich verschoben werden soll.
numberOfLines ForHMI	IN	UINT	O	1	Gibt die Anzahl an Zeilen (Meldungen) an, die für das HMI ausgegeben werden.
filterToHMI	IN	sLMsgHdl FilterToHMIType	O		Hier werden die Informationen übergeben, welche Meldungsquellen am Ausgang des FB angezeigt bzw. nicht angezeigt werden sollen. Ist im Meldungshandling die Verwendung von SIMOTION IT angewählt, werden diese Informationen in *gsLMsgHdlFilterToHMI* der Unit pLMsgHdl bereitgestellt und sollten als VAR_IN_OUT an den FB übergeben werden.
activeMessages BaseData	IN_OUT	sLMsgHdl Active Messages BaseDataType	M	-	Übergabe des aktuellen Meldepuffers im Format Rohdaten.
valid	OUT	BOOL	-	FALSE	Anzeige der Gültigkeit der Werte an den Ausgängen

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
error	OUT	BOOL	-	FALSE	Zeigt an, ob ein Fehler bei der Bearbeitung des FB aufgetreten ist.
errorId	OUT	DWORD	-	16#00000000	Gibt die Nummer des Fehlers zurück, der aufgetreten ist.
activeMsgToHMI	OUT	sLMsgHdlHMI ActiveMsg BaseDataType	-	-	Rückgabe der aktiven Meldungen aus dem Format Rohdaten für die Anzeige auf einem HMI.

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter, IN_OUT = Durchgangsparameter

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

Tabelle 5- 8 Fehlermeldungen

Fehlernummer [HEX]	Beschreibung
0	fehlerfrei
9997	Die Anzahl der im HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Anzahl der Zeilen für das HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in der Unit **cPublic**)
9998	Die Anzahl der am HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Länge des übergebenen Puffers (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES oder LMSGHDL_LENGTH_OF_MESSAGE_LOG in der Unit **cPublic**)

5.4.4 Struktur für Parameterübergabe

Die Struktur *sLMsgHdlHMIActiveMsgSgType* ist wie folgt aufgebaut.

Tabelle 5- 9 Struktur für aktive Meldungen als String an ein HMI

Parameter	Datentyp	Initialwert	Beschreibung
adtMessageOccured	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI -1] OF DT	DT#0001-01-01-0:0:0	Array für Informationen Meldung gekommen
ab32Parameter3	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI -1] OF DWORD	16#00000000	Array für Informationen in Parameter 3
ab32Parameter4	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI -1] OF DWORD	16#00000000	Array für Informationen in Parameter 4
ab32Parameter5	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI -1] OF DWORD	16#00000000	Array für Informationen in Parameter 5
ab32Parameter6	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI -1] OF DWORD	16#00000000	Array für Informationen in Parameter 6

Parameter	Datentyp	Initialwert	Beschreibung
ab32Parameter7	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 7
ab32Parameter8	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 8
ab32Parameter9	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 9
ab32Parameter10	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 10
ab32Parameter11	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 11
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI- 1] OF DINT	0	Array für Information, welche Nummer die entsprechende Meldung im Meldungspuffer für aktive Meldungen hat.
ai16Parameter2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF INT	0	Array für Informationen in Parameter 2
au8MessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Meldungsquelle
au8MessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Meldungslevel
au8AcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Quittierungsart
au8ErrorClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Fehlerklasse
au16Parameter1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF UINT	0	Array für Informationen in Parameter 1

5.5 Funktionsbaustein FBLMsgHdlMsgLogBaseDataToHMI

5.5.1 Allgemeines zum Funktionsbaustein

Zur Anzeige der Meldungstexte des globalen Puffers für MeldungsLog im Format Rohdaten an ein HMI kommt der Funktionsbaustein **FBLMsgHdlMsgLogBaseDataToHMI** zum Einsatz.

Verhalten wie in Funktionsbaustein FBLMsgHdlActiveMsgToHMI (Seite 73) beschrieben.

Siehe auch

Interpretation der Rohdaten (Seite 121)

5.5.2 Schematische Darstellung KOP/FUP

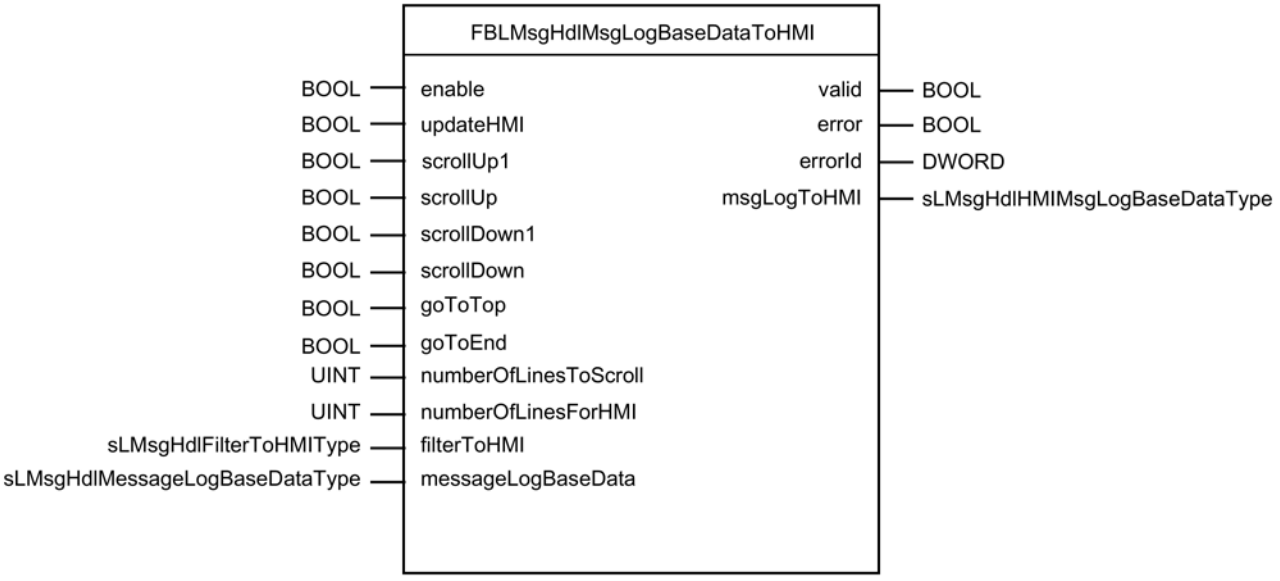


Bild 5-4 Schematische Darstellung KOP/FUP

5.5.3 Eingangs- und Ausgangsparameter des Bausteins

Der Funktionsbaustein **FBLMsgHdlMsgLogBaseDataToHMI** hat folgende Eingangs- und Ausgangsparameter:

Tabelle 5- 10 Eingangs- und Ausgangsparameter

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
enable	IN	BOOL	M	FALSE	Freigabe des FB
updateHMI	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der aktuell ausgegebene Datenbereich für HMI aktualisiert.
scrollUp1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach oben verschoben.
scrollUp	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in *numberOfLinesToScroll* übergebenen Wert nach oben verschoben
scrollDown1	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um eine Meldung nach unten verschoben
scrollDown	IN	BOOL	O	FALSE	Bei einer steigenden Flanke wird der anzuzeigende Datenbereich für HMI um den in *numberOfLinesToScroll* übergebenen Wert nach unten verschoben.
goToTop	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der obersten Zeile auf den aktuellsten Eintrag verschoben.
goToEnd	IN	BOOL	O	FALSE	Mit einer steigenden Flanke wird der Datenbereich zur Anzeige am HMI mit der untersten Zeile auf den ältesten Eintrag verschoben.
numberOfLines ToScroll	IN	UINT	O	1	Wert der vorgibt, um wie viele Zeilen bei Betätigung von *scrollUp,* bzw. *scrollDown* der Anzeigebereich verschoben wird.
numberOfLines ForHMI	IN	UINT	O	1	Gibt die Anzahl an Zeilen (Meldungen) an, die für das HMI ausgegeben werden.
filterToHMI	IN	sLMsgHdlFilter ToHMIType	O		Hier werden die Informationen übergeben, welche Meldungsquellen am Ausgang des FB angezeigt bzw. nicht angezeigt werden sollen. Ist im Meldungshandling die Verwendung von SIMOTION IT angewählt, werden diese Informationen in *gsLMsgHdlFilterToHMI* der Unit pLMsgHdl bereitgestellt und sollten als VAR_IN_OUT an den FB übergeben werden.
messageLog BaseData	IN_OUT	sLMsgHdl MessageLog BaseDataType	M	-	Übergabe des aktuellen Meldepuffers im Format Rohdaten.
valid	OUT	BOOL	-	FALSE	Anzeige der Gültigkeit der Werte an den Ausgängen
error	OUT	BOOL	-	FALSE	Zeigt an, ob ein Fehler bei der Bearbeitung des FB aufgetreten ist.

Bezeichnung	Typ [1]	Datentyp	M/O [2]	Initialwert	Beschreibung
errorId	OUT	DWORD	-	16#00000000	Gibt die Nummer des Fehlers zurück, der aufgetreten ist.
msgLogToHMI	OUT	sLMsgHdlHMI MsgLogBase DataType	-	-	Rückgabe der Meldungen aus dem Format Rohdaten für die Anzeige auf einem HMI.

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter, IN_OUT = Durchgangsparameter

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

Tabelle 5- 11 Fehlermeldungen

Fehlernummer [HEX]	Beschreibung
0	fehlerfrei
9997	Die Anzahl der im HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Anzahl der Zeilen für das HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in der Unit **cPublic**)
9998	Die Anzahl der am HMI anzuzeigenden Zeilen (Meldungen) im Parameter *numberOfLinesForHMI* ist größer als die maximale Länge des übergebenen Puffers (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES oder LMSGHDL_LENGTH_OF_MESSAGE_LOG in der Unit **cPublic**)

5.5.4 Struktur für Parameterübergabe

Die Struktur **sLMsgHdlHMIMsgLogBaseDataType** ist wie folgt aufgebaut.

Tabelle 5- 12 Struktur für MeldungsLog als String an ein HMI

Parameter	Datentyp	Initialwert	Beschreibung
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI- 1] OF DINT	0	Array für Information, welche Nummer die entsprechende Meldung im Meldungspuffer für aktive Meldungen hat.
au8MessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Meldungsquelle
au8MessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Meldungslevel
au8AcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Quittierungsart
au8ErrorClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array für Informationen zu Fehlerklasse
au16Parameter1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF UINT	0	Array für Informationen in Parameter 1

Parameter	Datentyp	Initialwert	Beschreibung
ai16Parameter2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF INT	0	Array für Informationen in Parameter 2
ab32Parameter3	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 3
ab32Parameter4	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 4
ab32Parameter5	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 5
ab32Parameter6	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 6
ab32Parameter7	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 7
ab32Parameter8	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 8
ab32Parameter9	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 9
ab32Parameter10	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 10
ab32Parameter11	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array für Informationen in Parameter 11
adtMessageOccured	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DT	DT#0001 -01-01- 0:0:0	Array für Informationen Meldung gekommen
adtMessageGone	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DT	DT#0001 -01-01- 0:0:0	Array für Informationen Meldung gegangen

5.6 Funktionen FCLMsgHdlWriteUserMessageToBuffer und FCLMsgHdlWriteFBFCMessageToBuffer

5.6.1 Allgemeines zu den Funktionen

Durch den Aufruf der Funktion **FCLMsgHdlWriteUserMessageToBuffer** bzw. **FCLMsgHdlWriteFBFCMessageToBuffer** werden anwenderdefinierte Meldungen an das Meldungshandling übergeben. Diese Meldungen werden aus der Applikation generiert. Für jede neue Meldung muss diese Funktion einmal aufgerufen werden. Bei Meldungen, die bereits aktiv sind und noch nicht quittiert wurden, wird keine neue Meldung in die Puffer eingetragen bzw. eine Meldung über Bitmeldeverfahren oder AlarmS ausgelöst.

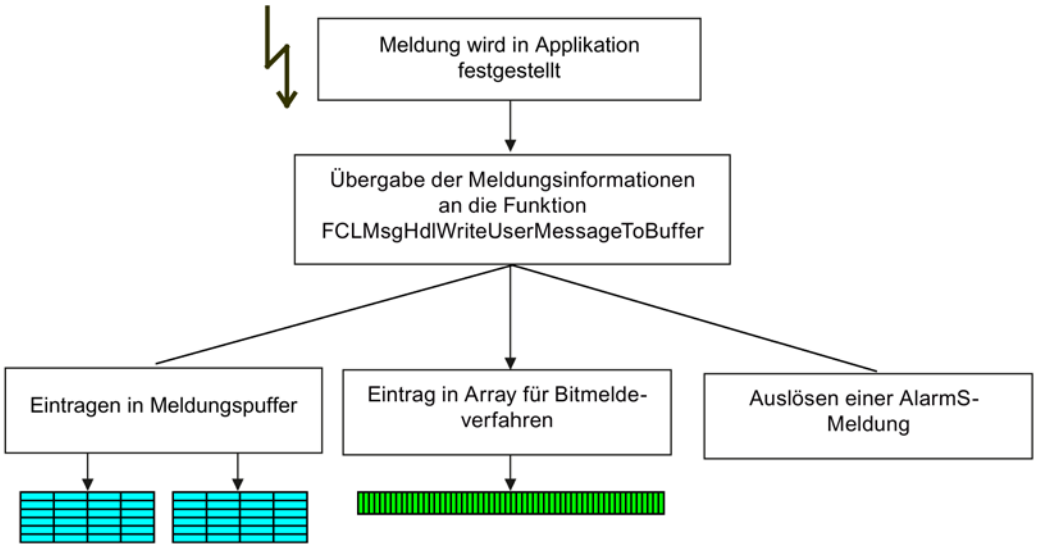


Bild 5-5 Eintragen von anwenderdefinierten Meldungen

Eine Meldung wird über ihre Meldungsnummer eindeutig vorgegeben. Die eingetragenen Meldungen können auf drei verschiedene Arten für die Weitermeldung an den Bediener verarbeitet werden.

- Eintrag in die Meldepuffer:
  Die Meldung wird in die Meldepuffer eingetragen. Der Anwender hat dafür zu sorgen, dass bei Verwendung der stringbasierten Puffer, ein Meldungstext in der Steuerung hinterlegt wird, siehe Abschnitt Integration der Applikation in ein SIMOTION Projekt (Seite 55).
- Verwendung des Bitmeldeverfahrens:
  Bei Anwahl des Bitmeldeverfahrens, durch Setzen der Konstante LMSGHDL_MESSAGE_BIT_USER_MESSAGES auf TRUE, wird das der Meldungsnummer entsprechende Bit im globalen WORD-Array *gab16LMsgHldEventFlag* in der Unit **fLMsgHdl** gesetzt. Die Meldungstexte werden im HMI hinterlegt.
- Verwendung von AlarmS:
  Bei Anwahl des AlarmS-Meldungsverfahrens, durch Setzen der Konstante LMSGHDL_ALARM_S_USER_MESSAGES auf TRUE wird eine der Meldungsnummer

entsprechende AlarmS-Meldung generiert. Die Texte der Meldungen sind vom Anwender in SIMOTION SCOUT einzugeben.

Aufrufbeispiel

```
FCLMagHdlWriteFCFBMessageToBuffer(eventNumber         := 8
                                 ,errorClass          := 2
                                 ,errorCode           := 16#ffff8082
                                 ,functionBlockId     := 1
                                 ,additionalValue1DINT := 512
                                 ,additionalValue2DINT := 4711);
```

Die Eventnummer (*eventNumber*) bestimmt die anwenderdefinierte Meldung, die im Meldungshandling ausgegeben wird. Gleichzeitig wird bei aktivem AlarmS oder Bitmeldeverfahren die entsprechende Meldung gesetzt. Ist keine zusätzliche *functionBlockId* gesetzt, wird die zur Eventnummer gehörende Meldungsklasse ausgelöst.

Die *functionBlockId* wird gesetzt, falls es sich um eine anwenderdefinierte Meldung durch eine Funktion oder einen Funktionsbaustein handelt. In diesem Fall wird nicht die Meldungsklasse aus der Eventnummer gebildet, sondern aus der entsprechenden Vorgabe der Variablen *gasLMsgHdlFBFCMachineErrorClasses*. Hierbei wird der Wert für die Maschinenfehlerklasse wie folgt ausgelesen und ins Meldungshandling übernommen: `gasLMsgHdlFBFCMachineErrorClasses[0].ai8ErrorClass[2]`

5.6.2 Schematische Darstellung KOP/FUP

Funktion FCLMsgHdlWriteUserMessageToBuffer

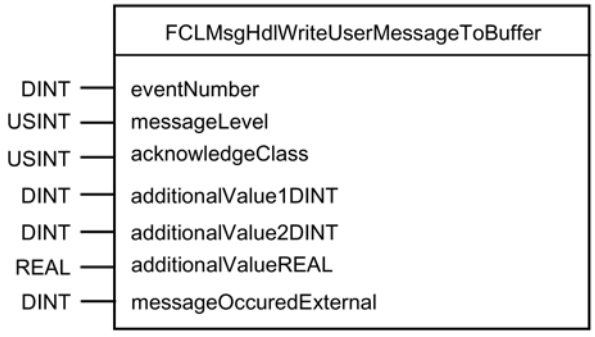


Bild 5-6 Schematische Darstellung KOP/FUP

Funktion FCLMsgHdlWriteFBFCMessageToBuffer

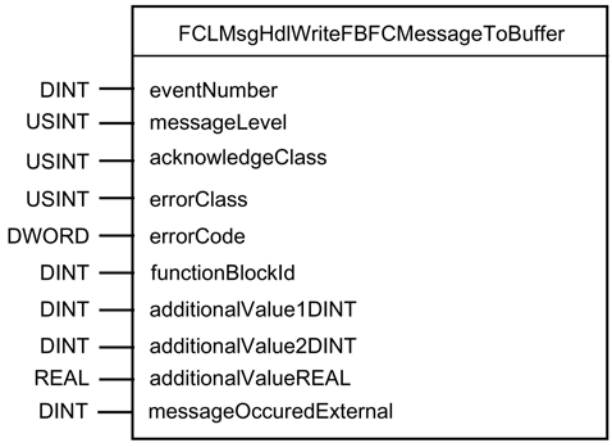


Bild 5-7 Schematische Darstellung KOP/FUP

5.6.3 Eingangs- und Ausgangsparameter der Funktionen

Der Anwender muss bei den Meldungsnummern mit 1 (Eins) beginnen, die 0 (Null) ist nicht zulässig.

Eingangs- und Ausgangsparameter der Funktion FCLMsgHdlWriteUserMessageToBuffer

Die Funktion **FCLMsgHdlWriteUserMessageToBuffer** hat folgende Eingangs- und Ausgangsparameter.

Tabelle 5- 13 Eingangs- und Ausgangsparameter

Element	P-Type [1]	Datentyp	M/O [2]	Initialwert	Bedeutung
eventNumber	IN	DINT	M	0	Meldungsnummer
messageLevel	IN	USINT	O	2	Übergabe des Meldungslevels der Meldung (Default ist 2 -> Fehler)
acknowledgeClass	IN	USINT	O	2	Übergabe der Art der Quittierung (Default ist 2 -> direkt)
additionalValue1DINT	IN	DINT	O	0	zusätzlicher Beiwert zum Fehler im Format DINT, wird an AlarmS übergeben
additionalValue2DINT	IN	DINT	O	0	zusätzlicher Beiwert zum Fehler im Format DINT
additionalValueREAL	IN	REAL	O	0.0	Zusätzlicher Beiwert um Fehler im Format REAL zu übergeben

Element	P-Type [1]	Datentyp	M/O [2]	Initialwert	Bedeutung
messageOccuredExternal	IN	DINT	O	0	Wenn dieser Eingang verwendet wird, wird der dort übergebene Wert an das Meldungshandling übergeben. Wenn über diesen Eingang kein Wert übergeben wird, gilt die Systemuhrzeit des SIMOTION Geräts. Voreinstellung ist die Systemuhrzeit des SIMOTION Geräts.
FCLMsgHdlWriteMessageToBuffer	OUT	VOID	-	-	kein Rückgabecode der Funktion

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter,

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

Eingangs- und Ausgangsparameter der Funktion FCLMsgHdlWriteFBFCMessageToBuffer

Die Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** hat folgende Eingangs- und Ausgangsparameter.

Tabelle 5- 14 Eingangs- und Ausgangsparameter

Element	P-Type [1]	Datentyp	M/O [2]	Initialwert	Bedeutung
eventNumber	IN	DINT	M	0	Meldungsnummer
messageLevel	IN	USINT	O	2	Übergabe des Meldungslevels der Meldung (Default ist 2 -> Fehler)
acknowledgeClass	IN	USINT	O	2	Übergabe der Art der Quittierung (Default ist 2 -> direkt)
errorClass	IN	USINT	O	0	Fehlerklasse der Meldung
errorCode	IN	DWORD	O	16#00000000	Fehlercode (z. B. bei Fehlerrückmeldung von Funktionsbausteinen), wird an AlarmS übergeben
functionBlockId	IN	DINT	O	0	ID des meldungsauslösenden Funktionsbausteins
additionalValue1DINT	IN	DINT	O	0	zusätzlicher Beiwert zum Fehler im Format DINT
additionalValue2DINT	IN	DINT	O	0	zusätzlicher Beiwert zum Fehler im Format DINT
additionalValueREAL	IN	REAL	O	0.0	Zusätzlicher Beiwert um Fehler im Format REAL zu übergeben
messageOccuredExternal	IN	DINT	O	0	Wenn dieser Eingang verwendet wird, wird der dort übergebene Wert an das Meldungshandling übergeben. Wenn über diesen Eingang kein Wert übergeben wird, gilt die Systemuhrzeit des SIMOTION Geräts. Voreinstellung ist die Systemuhrzeit des SIMOTION Geräts.
FCLMsgHdlWriteFCFBMessageToBuffer	OUT	VOID	-	-	kein Rückgabecode der Funktion

[1] Parametertypen: IN = Eingangsparameter, OUT = Ausgangsparameter,

[2] Parameterart: M = Pflichtparameter, O = Optionaler Parameter

5.7 Struktur für MeldungsLog als Rohdaten

Tabelle 5- 15 Struktur für das MeldungsLog als Rohdaten *sLMsgHdlMessageLogBaseDataType*

Parameter	Datentyp	Initialwert	Beschreibung
i16ActualIndex	INT	0	Aktueller Index im globalen MeldungsLog für Rohdaten
i16NumberOfNewMessages	INT	0	Nur für internen Gebrauch von Meldungshandling
boChangesInLogBaseData	BOOL	FALSE	Nur für internen Gebrauch von Meldungshandling
boNewMessagesForHMILogBase	BOOL	FALSE	Nur für internen Gebrauch von Meldungshandling
boBuildNewStringMessages	BOOL	FALSE	Nur für internen Gebrauch von Meldungshandling
au8MessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF USINT	0	Array für Information über Quelle der Meldung
au8MessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF USINT	0	Array für Information über Stufe der Meldung (Störung, Warnung, Fehler, Information)
au8AcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF USINT	0	Array für Information über die Quittierungsart der Meldung
au8ErrorClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF USINT	0	Array für Information über Fehlerklasse einer Meldung
au16Parameter1	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF UINT	0	Array für Information Variable1 des jeweiligen Meldungstyps
ai16Parameter2	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF INT	0	Array für Information Variable2 des jeweiligen Meldungstyps
ab32Parameter3	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable3 des jeweiligen Meldungstyps
ab32Parameter4	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable4 des jeweiligen Meldungstyps
ab32Parameter5	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable5 des jeweiligen Meldungstyps
ab32Parameter6	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable6 des jeweiligen Meldungstyps
ab32Parameter7	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable7 des jeweiligen Meldungstyps
ab32Parameter8	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable8 des jeweiligen Meldungstyps
ab32Parameter9	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable9 des jeweiligen Meldungstyps
ab32Parameter10	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable10 des jeweiligen Meldungstyps
ab32Parameter11	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DWORD	16#00000000	Array für Information Variable11 des jeweiligen Meldungstyps
adtMessageOccured	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG - 1] OF DT	DT#0001-01-01-0:0:0:0	Zeitstempel, wann die entsprechende Meldung aufgetreten ist.

Parameter	Datentyp	Initialwert	Beschreibung
adtMessageGone	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DT	DT#0001-01-01-0:0:0:0	Zeitstempel, wann die entsprechende Meldung gegangen ist.
boNewMessage	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] BOOL	FALSE	Zeigt an, ob die entsprechende Meldung neu ins MeldungsLog Rohdaten eingetragen wurde.
i16MessageIndexLast Cycle	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] INT	0	Index der Meldung im vorherigen Backgroundzyklus. Hilfsvariable für Update des MeldungsLog im Format String.

5.8 Struktur für MeldungsLog als String

Tabelle 5- 16 Struktur für das MeldungsLog als String *sLMsgHdlMessageLogStringType*

Parameter	Datentyp	Initialwert	Beschreibung
i16ActualIndex	INT	0	Aktueller Index im globalen MeldungsLog für String.
boChangesIn LogStringData	BOOL	FALSE	Nur für internen Gebrauch von Meldungshandling
boChangesIn LogStringDataForHMI	BOOL	FALSE	Nur für internen Gebrauch von Meldungshandling
asgMessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[11]	Leerstring	Array für Information über Stufe der Meldung (Störung, Warnung, Fehler, Information).
asgMessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[64]	Leerstring	Array für Information über Quelle der Meldung.
asgMessageText	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[160]	Leerstring	Array für sprachabhängigen Meldungstext.
asgMessageOccured	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[23]	Leerstring	Zeitstempel, wann die entsprechende Meldung aufgetreten ist.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[17]	Leerstring	Array für Information über die Quittierungsart der Meldung.
asgMessageGone	ARRAY[0.LMSGHDL_LENGTH_OF_ MESSAGE_LOG- 1] OF STRING[23]	Leerstring	Zeitstempel, wann die entsprechende Meldung gegangen ist.

Alarm- und Fehlermeldungen 6

6.1 Allgemeines zum Fehlerhandling

Im Meldungshandling selbst können auch Fehler bei der Bearbeitung auftreten. Diese werden gemeldet, wenn Fehler bei der Abarbeitung des Meldungshandlings auftreten, z. B. fehlgeschlagene Kommunikation mit Antriebsobjekten eines SINAMICS Antriebs.

6.2 Überlauf von Puffern

Die Meldungen der einzelnen möglichen Meldungsquellen werden zunächst in Zwischenpuffern gesammelt. Damit wird die Datenkonsistenz der Meldungsinformationen gewährleistet. Diese Zwischenpuffer sind als Ringpuffer ausgelegt und geben die durch eine neue Meldung beschriebenen Speicherbereiche erst dann wieder zum neuen Beschreiben frei, wenn diese Informationen in die globalen Meldungspuffer eingetragen wurden.

Treten an einer Meldungsquelle sehr viele neue Meldungen in einer kurzen Zeit auf, kann es vorkommen, dass ein Zwischenpuffer schneller beschrieben, als ausgelesen wird. Geschieht dies, kann es zum Überlauf eines Zwischenpuffers kommen. Der Überlauf eines dieser Zwischenpuffer wird durch eine interne Fehlermeldung an das Meldungshandling übergeben.

Wurde eine solche Meldung eingetragen, kann nicht mehr sichergestellt werden, dass die Informationen in den Meldepuffern vollständig sind. Alle Meldungen einer Quelle, die nach einem Überlauf auftreten, werden nicht in die Meldungspuffer übernommen und gehen verloren. Erst nachdem über das Meldungshandling eine Quittierung stattgefunden hat, werden an der betroffenen Quelle neue Meldungen wieder erkannt.

Der betroffene Zwischenpuffer kann über die jeweilige Konstante in **cPublic** der Bibliothek **LMsgHdl** vergrößert werden. Dieser Wert ist jedoch erst nach einer neuen Übersetzung und einem Download des Projekts gültig.

6.3 Überlauf AlarmS-Meldungen

Bei Verwendung von AlarmS im Meldungshandling ist darauf zu achten, dass nur 40 AlarmS-Meldungen gleichzeitig aktiv sein dürfen. Wird die nächste Meldung ausgelöst, kann diese nicht mehr vom AlarmS-Handler bearbeitet werden und geht verloren. Die Meldungen werden trotzdem ins Meldungshandling übernommen. Gleichzeitig wird im Meldungshandling der interne Fehler **LMSGHDL_ALARM_S_ERROR (100002)** ausgelöst und ins Meldungshandling eingetragen.

6.4 Fehler beim Hochlauf

Das Meldungshandling überwacht den Hochlauf der Maschine und sammelt dabei generisch Informationen, die im weiteren Verlauf im Meldungshandling benötigt werden. Sind nicht alle projektierten Baugruppen an der Steuerung vorhanden, kann nicht sichergestellt werden, dass das Meldungshandling ordnungsgemäß arbeitet. Aus diesem Grund wird vom Meldungshandling ein interner Fehler gemeldet, sobald eine projektierte Baugruppe nicht vorhanden ist oder einen anderen Betriebszustand als aktiv meldet **LMSGHDL_ERROR_CHECK_STARTUP_OK (100026)**.

Der Zeitüberlauf (Timeout) läuft ab, wenn bei einer SIMOTION D keine azyklische Kommunikation zum SINAMICS_Integrated möglich ist.

Nach der Meldung einer Stationswiederkehr einer Peripheriebaugruppe wird die Prüfung, ob die Peripheriebaugruppe wieder vorhanden ist, automatisch neu gestartet.

Tritt im Hochlauf ein Fehler in der Pufferverwaltung auf, wird die Hochlaufprüfung sofort beendet und die Meldung über den Fehler der Pufferverwaltung an das Meldungshandling übergeben. Meldet die Pufferverwaltung einen Fehler, kann das Meldungshandling erst nach einem Neustart wieder verwendet werden.

6.5 Meldungen durch Peripheriebaugruppen

Jede Peripheriebaugruppe hat in der Variablen *gasLMsgHdlPeripheralDevices* der Bibliotheksunit *dProtected* eine Variable *boStationConnected* vom Typ BOOL welches aussagt, ob die Peripheriebaugruppe vorhanden ist oder nicht. Das Meldehandling setzt dieses Bit bei Stationsausfall bzw. Stationswiederkehr.

6.6 DO-Safety-Meldungen

Safety-Meldungen werden wie Fehler am Antriebsobjekt gehandhabt. Safety-Meldungen können nicht vom Meldungshandling quittiert werden. Aus diesem Grund wird der Zeitstempel **Meldung gegangen** überwacht. D. h., dass jede Safety-Meldung beim Auftreten und beim Quittieren einen Eintrag im Zwischenpuffer für Safety-Meldungen an allen Antriebsobjekten erzeugt. Im Meldungshandling muss darauf geachtet werden, dass aktive Safety-Meldungen eigenständig beim Quittieren wieder aus dem Puffer für aktive Meldungen herausgenommen werden. Zusätzlich muss im globalen **MeldungsLog** eine aktive Safety-Meldung beim Verschwinden lediglich mit dem Zeitstempel für **Meldung gegangen** ergänzt werden.

Die Zwischenpufferung für Safety-Meldungen aus der vorherigen Detektion stehen in der Variable *gasLMsgHdlAuxiliaryBufferDOWithTOSafety.* Die neu gefundenen Safety-Meldungen werden im Array *asLMsgHdlAuxiliaryBufferDOSafety* abgelegt.

6.7 Anwenderdefinierte Meldungen

Im Meldungshandling ist es möglich, für alle anwenderdefinierten Meldungen einen eigenständig erzeugten Zeitstempel, wann die Meldung aufgetreten ist zu übergeben. Hierfür gibt es an allen FC den Eingang *messageOccuredExternal* vom Typ DT. Wird hier ein Wert ungleich des Defaultwerts übergeben, wird anstelle der aktuellen Systemzeit, dieser Zeitstempel ans Meldungshandling übergeben. Wird dieser Eingang nicht befüllt oder mit einem Defaultwert beschrieben, übernimmt das Meldungshandling den aktuellen Systemzeitstempel.

Es besteht die Möglichkeit die anwenderdefinierte Meldung mit der Variablen *EventID = 0* zu erzeugen. Mit Vergabe dieser Variablen als anwenderdefinierte Meldung ist es möglich, eine Meldung mehrfach zu erzeugen. Alle anderen Meldungen können jeweils nur einmalig aktiv sein. Wird die *EventId = 0* vergeben, wird die Meldung mit jedem neuen Aufruf mit dem jeweiligen Zeitstempel erneut ins Meldungshandling eingetragen. Diese Meldung erzeugt jedoch keine Meldungsklasse, keinen zugehörigen AlarmS und kein Bit im Bitmeldeverfahren. Der zur STRING Ausgabe benötigte Meldungstext steht in der Unit **fLMsgHdlInit** in der Variablen *gasLMsgHdlUserDefinedMessageEvent0*. Die anwenderdefinierte Meldung mit *EventId = 0* kann über die Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** erzeugt werden. Somit ist es möglich, dieser Meldung als Beiwerte einen Fehlercode, eine *functionBlockId* und zwei Beiwerte zu übergeben. Die so erzeugten anwenderdefinierten Meldungen lassen sich ebenfalls einzeln quittieren.

6.8 Fehler bei Kommunikation mit DOs

Tritt ein Fehler bei der Kommunikation zu mindestens einem DO auf, wird dem Meldungshandling eine entsprechende Meldung übergeben. Die Kommunikation mit dem entsprechenden DO wird anschließend unterbrochen. Die Kommunikation zum DO wird erst nach einer, durch das Meldungshandling ausgelösten globalen Quittierung, wieder gestartet. Tritt der Fehler in der Kommunikation erneut auf, wird wieder eine Meldung ausgegeben.

Das DO, das die Meldung verursacht, wird je nach Typ des DO, entweder über die Nummer der zugehörigen Achse, der logischen Adresse oder einer logischen Adresse inklusive der DO-Nummer ans Meldungshandling übergeben.

6.9 Besonderheit bei Warnungen an Antriebsobjekten

Warnungen von Antriebsobjekten auf SINAMICS-Baugruppen können nicht quittiert werden. Treten diese auf, bleiben sie solange anstehen bis der Grund der Warnung nicht mehr besteht. Sobald eine aktive Warnung nicht mehr ansteht, wird sie automatisch aus den Meldepuffern für aktive Meldungen gelöscht.

6.10 Besonderheit bei Peripherie-Meldungen

Meldungen durch Peripheriebaugruppen haben in SIMOTION unterschiedlichen Charakter. Es gibt **negative** und **positive** Peripheriemeldungen. Zu den negativen gehört z. B. die Meldung *Stationsausfall*. Dagegen gibt es die positive Meldung *Stationswiederkehr*. Das Sammeln dieser Peripheriemeldungen ist im Meldungshandling daher wie folgt umgesetzt worden.

Tritt eine negative Peripheriemeldung an einer Maschine auf, wird diese in die Meldungshistorie und den Meldepuffer für aktive Meldungen eingetragen. Positive Meldungen werden nur in die Meldungshistorie übernommen. Gleichzeitig wird der Meldungspuffer für aktive Meldungen daraufhin untersucht, ob noch eine zur positiven Meldung gehörende negative Meldung aktiv ist. Ist dies der Fall, wird die entsprechende Meldung automatisch quittiert und aus den aktiven Meldungen herausgenommen. Wird eine negative aktive Meldung aus Peripherie vor dem Auftreten der zugehörigen positiven Meldung quittiert, wird sie ebenfalls aus dem Puffer für aktive Meldungen herausgenommen. Das Auftreten aller Peripheriemeldungen ist demnach immer im Historienspeicher des Meldungshandlings nach zu verfolgen.

Tabelle 6- 1 Zueinander gehörenden positiven und negativen Peripheriemeldungen

Negative Meldung	Positive Meldung
ID 202 Stationsausfall	ID 203 Stationswiederkehr
ID 204 Fehler beim Bilden des Prozessabbildes	ID 206 Bilden des Prozessabbildes funktioniert wieder
ID 210 Mehrfacher Taktausfall oder PLL ausgerastet	ID209 PLL in geregelten Betrieb eingerastet
ID 215 Synchronisation ausgefallen	ID214 Synchronisation erreicht

Die Peripheriemeldungen mit den Interrupt-IDs 200, 201, 205, 208, 211, 212, 213, 216 und 217 werden ebenfalls in den Historienspeicher und den Meldepuffer für aktive Meldungen übernommen. Diese IDs haben jedoch keine komplementäre Peripheriemeldung und werden nicht automatisch quittiert. Die Bedeutung der hier aufgeführten IDs sind in der SIMOTION SCOUT Online-Hilfe unter **Taskstartinfo verwenden** beschrieben.

6.11 Reaktion auf eigene Fehler

Tritt im Meldungshandling ein Fehler auf, so wird dieser über eine entsprechende Meldung in die jeweiligen Meldungspuffer eingetragen. Zusätzlich wird eine globale Maschinenfehlerklasse gesetzt. Diese Maschinenfehlerklasse ist für alle Meldungen des Meldungshandlings identisch und kann vom Anwender in der Unit **cPublic** der Bibliothek **LMsgHdl** über die Konstante LMSGHDL_MACHINE_ERROR_CLASS_ERROR_IN_MESSAGEHANDLING festgelegt werden.

Die durch interne Fehler erzeugten Meldungen können über die globale Quittierung zurückgesetzt werden. Ist der Grund für den internen Fehler jedoch zum Zeitpunkt der Quittierung nicht behoben, wird die Meldung erneut ins Meldungshandling eintragen. Da es sich bei internen Fehlern im Meldungshandling um anwendungsspezifische Meldungen handelt, werden diese identisch zu den anwenderdefinierten Meldungen an das

Meldungshandling übergeben. Um eine klare Trennung zu den Anwendermeldungen zu erreichen, beginnen diese Eventnummern bei 100.000.

Folgende interne Fehler können im Meldungshandling auftreten:

Tabelle 6- 2 Interne Fehler im Meldungshandling

Name der Konstante	Event-nummer	Bedeutung
LMSGHDL_UNKNOWN_USER_EVENT		
	100.000	Es wurde eine unbekannte anwenderdefinierte Meldung ans Meldungshandling übergeben. Anhand der Beiwerte kann die fehlerhafte Meldung identifiziert werden. Diese Meldung kann nur quittiert werden.
LMSGHDL_EVENT_IN_UNKNOWN_TASK		
	100.001	Es wurde eine anwenderdefinierte Meldung aus einer dem Meldungshandling unbekannten Task heraus übergeben. Eine Änderung des Ablaufsystems ohne erneutes Aufrufen des Skripts wurde vorgenommen. Das Konfigurations-Skript sollte nochmals durchlaufen werden.
LMSGHDL_ALARM_S_ERROR		
	100.002	Beim Auslösen einer AlarmS-Meldung ist ein Fehler aufgetreten. Die intern verwendeten Funktionen *_alarmSQId* bzw. *_alarmSId* haben einen Fehler erzeugt. Behebung: siehe Beiwert *errorId*. (Fehler der Systemfunktionen). Diese Meldung kann sofort quittiert werden.
LMSGHDL_OVERFLOW_BUFFER_EXECUTIONTASK_MESSAGES		
	100.003	Der Zwischenpuffer zum Sammeln der ExecutionTask-Meldungen hat einen Überlauf. Es sind in zwei aufeinanderfolgenden Starts der Maschine ExecutionFault-Meldungen aufgetreten, ohne dass das Meldungshandling die erste Meldung nach Neuanlauf verarbeiten konnte. Suche des Fehlers und Neustart der Maschine. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_OVERFLOW_BUFFER_APPLICATION_MESSAGES		
	100.004	Der Zwischenpuffer zum Sammeln der Anwender- Meldungen hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_APPLICATION_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_OVERFLOW_BUFFER_TECHFAULT_MESSAGES		
	100.005	Der Zwischenpuffer zum Sammeln der TechnologicalFaultTask-Meldungen hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_PERIPHERAL_FAULT_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_OVERFLOW_BUFFER_PERIPHERAL_MESSAGES		
	100.006	Der Zwischenpuffer zum Sammeln der PeripheralFaultTask-Meldungen hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_TECH_FAULT_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_OVERFLOW_BUFFER_TIMEFAULT_MESSAGES		
	100.007	Der Zwischenpuffer zum Sammeln der TimeFaultTask-Meldungen hat einen Überlauf, z. B. bei einer Endlosschleife in der BackgroundTask. Die Konstante LMSGHDL_NUMBER_OF_TIME_FAULT_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.

Name der Konstante	Event-nummer	Bedeutung
LMSGHDL_OVERFLOW_BUFFER_DOFAULT_MESSAGES		
	100.008	Der Zwischenpuffer zum Sammeln der Fehler an allen Driveobjekten hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_DO_FAULT_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_OVERFLOW_BUFFER_DOALARM_MESSAGES		
	100.009	Der Zwischenpuffer zum Sammeln der Warnungen an allen Driveobjekten hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_DO_ALARM_MESSAGES in **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_ERROR_IN_BUFFER_MANAGER		
	100.010	In der Verwendung der Pufferverwaltung ist ein Fehler aufgetreten. Da die Pufferverwaltung für DP-V1-Dienste einen Fehler hat, werden die SINAMICS Antriebsobjekte nicht mehr überwacht. Ein Neustart der Maschine ist erforderlich.
LMSGHDL_ERROR_IN_CHECK_STARTUP		
	100.011	Im Funktionsbaustein **FBLDPV1CheckStartup** für die Hochlaufkontrolle ist ein Fehler aufgetreten. Nicht alle projektiert en Peripheriebaugruppen sind vorhanden. Das Meldungshandling kann nicht alle notwendigen Informationen bereitstellen. Eine Überprüfung der Maschine und eventuell Neustart sind notwendig.
LMSGHDL_ERROR_ADD_TO		
	100.012	Es wurde versucht eine nicht existierende Achse ans Meldungshandling zu übergeben. Ein neuer Durchlauf des Konfigurations-Skripts ist erforderlich.
LMSGHDL_ERROR_SEARCH_ALL_DOS		
	100.013	Bei der internen Zuordnung von Technologieobjekten zu Driveobjekten ist ein Fehler aufgetreten, ausgelöst durch die Funktion **FBLDPV1SearchAllDo**. Siehe Übergabeparameter *errorId* und Dokumentation DO-TO in LDPV1. Das Meldungshandling kann nicht alle notwendigen Informationen bereitstellen. Eine Überprüfung der Maschine und eventuell Neustart sind notwendig.
LMSGHDL_ERROR_CHECK_WRITE_ACCESS		
	100.014	Bei der Kontrolle ob Parameter in SINAMICS Driveobjekten geschrieben werden dürfen, ist ein Fehler aufgetreten. Siehe Übergabeparameter *errorId* und Dokumentation *CheckStartup* in LDPV1. Überprüfung der Maschine und eventuell Neustart sind notwendig.
LMSGHDL_ERROR_TIME_SYNC		
	100.015	Bei der Uhrzeitsynchronisation einer SINAMICS Baugruppe ist ein Fehler aufgetreten. Siehe Übergabeparameter *errorId* und Dokumentation *TimeSync* in LDPV1. Meldungen von Antriebsobjekten haben eventuell nur den Zeitstempel der SIMOTION RTC und können nicht richtig zugeordnet werden. Überprüfung der Maschine und eventuell Neustart sind erforderlich.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_AT_AXIS		
	100.016	Bei der Ermittlung von Fehlern/Warnungen an einem Driveobjekt, das zu einem Achs-Technologieobjekt gehört, ist ein Fehler aufgetreten. Siehe Übergabeparameter *errorId* und Dokumentation *GetFault* in LDPV1. Eventuell ist eine SINAMICS Baugruppe komplett ausgefallen. Eine Überprüfung der Maschine ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_CYCLIC_DOS		

Name der Konstante	Event-nummer	Bedeutung
	100.017	Bei der Ermittlung von Fehlern/Warnungen an einem Driveobjekt mit zyklischer Kommunikation ist ein Fehler aufgetreten. Siehe Übergabeparameter *errorId* und Dokumentation *GetFault* in LDPV1. Eventuell ist eine SINAMICS Baugruppe komplett ausgefallen. Eine Überprüfung der Maschine ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_ACYCLIC_DOS		
	100.018	Bei der Ermittlung von Fehlern/Warnungen an einem Driveobjekt ohne zyklische Kommunikation ist ein Fehler aufgetreten. Siehe Übergabeparameter *errorId* und Dokumentation *GetFault* in LDPV1. Eventuell ist eine SINAMICS Baugruppe komplett ausgefallen. Eine Überprüfung der Maschine ist notwendig. Eine Sofortige Quittierung ist möglich.
LMSGHDL_ERROR_MESSAGE_BUFFER_MANAGER		
	100.019	Im Baustein zur Erstellung der Meldungspuffer ist ein Fehler aufgetreten (Acknowledge der anstehenden Meldungen). Eventuell ist eine SINAMICS Baugruppe komplett ausgefallen. Eine Überprüfung der Maschine ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_GET_DO_NAME		
	100.020	Fehler beim selbstständigen Ermitteln der Namen aller projektierten Driveobjekte. Das Meldungshandling kann nicht alle notwendigen Informationen bereitstellen. Eventuell ist eine SINAMICS Baugruppe komplett ausgefallen. Eine Überprüfung der Maschine und eventuell Neustart sind notwendig.
LMSGHDL_ERROR_INIT_MESSAGELOG_STRING		
	100.021	Fehler beim Zusammenstellen der Meldungsinformationen im Format String. Eine Kontrolle der im System hinterlegten Texte im Format String ist notwendig.
LMSGHDL_ERROR_UPDATE_HMI_ACTIVE_MESSAGES		
	100.022	Fehler im Baustein zur Ausgabe der aktiven Meldungen im Format String an HMI, bzw. SIMOTION IT. Eine Kontrolle der Übergabeparameter für die Ausgabe an ein HMI ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_UPDATE_HMI_MESSAGE_LOG		
	100.023	Fehler im Baustein zur Ausgabe der Meldungshistorie im Format String an HMI, bzw. SIMOTION IT. Eine Kontrolle der Übergabeparameter für die Ausgabe an ein HMI ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_CHANGE_LANGUAGE_SYSTEM		
	100.024	Ausgewählte Sprache als Datei für Systemmeldungen ist nicht auf dem Speichermedium des SIMOTION Gerätes vorhanden oder fehlerhaft. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_CHANGE_LANGUAGE_USER		
	100.025	Ausgewählte Sprache als Datei für anwenderdefinierte Meldungen ist nicht auf dem Speichermedium des SIMOTION Gerätes vorhanden oder fehlerhaft. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_CHECK_STARTUP_OK		
	100.026	Nicht alle projektierten Geräte wurden als betriebsbereit erkannt. Das Meldungshandling kann nicht alle notwendigen Informationen bereitstellen. Eventuell ist eine Peripheriebaugruppe komplett ausgefallen. Eine Überprüfung der Maschine und eventuell Neustart sind notwendig.
LMSGHDL_ERROR_CHANGE_LANGUAGE		

Name der Konstante	Event-nummer	Bedeutung
	100.027	Fehler beim Wechseln der aktiven Sprache zur Ausgabe der Meldungen im Format String. Eine Kontrolle der Sprachdateien auf dem Speichermedium ist notwendig. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_WRITE_MESSAGE_LOG_TO_STORAGE_MEDIUM		
	100.028	Fehler beim Schreiben ins MeldungsLog auf das Speichermedium des SIMOTION Geräts. Eventuell sind nicht genug Systemressourcen verfügbar. Eine sofortige Quittierung ist möglich.
LMSGHDL_OVERFLOW_BUFFER_DOSAFETY_MESSAGES		
	100.029	Der Zwischenpuffer zum Sammeln der Safety-Meldungen an allen Antriebsobjekten hat einen Überlauf. Die Konstante LMSGHDL_NUMBER_OF_DO_SAFETY_MESSAGES in der Unit **cPublic** muss vergrößert werden. Eine Quittierung ist möglich. Die auslösende Meldung konnte nicht übernommen werden.
LMSGHDL_ILLEGAL_FUNCTION_BLOCK_ID		
	100.030	Beim Aufruf einer anwenderdefinierten Meldung durch FB / FC wurde eine unerlaubte *functionBlockId* übergeben. Eine sofortige Quittierung ist möglich.
LMSGHDL_ILLEGAL_ERROR_CLASS		
	100.031	Beim Aufruf einer anwenderdefinierten Meldung durch FB / FC wurde eine unerlaubte *errorClass* übergeben. Eine sofortige Quittierung ist möglich.
LMSGHDL_ERROR_IN_PERSITENT_DATA_POWER_MONITORING		
	100.032	Es ist ein Fehler bei der Spannung für die Sicherung der Netzausfesten-Daten aufgetreten.

Anwendungsbeispiel 7

7.1 Maschinenfehlerklassen definieren (Beispiel)

Dieses Anwendungsbeispiel zeigt die wichtigsten Anpassungen eines SIMOTION Projektes. Die Programmierung der Meldungen hängt von den Anforderungen des Anwenders ab. Dieses Beispiel kann deshalb nur Ansätze der Umsetzung aufzeigen.

Im diesem Anwendungsbeispiel wurde Folgendes definiert:

- 10 Maschinenfehlerklassen
- 10 anwenderdefinierte Meldungen
- Meldungen für 3 FBs / FCs mit jeweils 4 Fehlerklassen
- Meldungen für 2 Peripherie Geräte (Antrieb SINAMICS Integrated und Control Unit CU310)

Maschinenfehlerklassen editieren

In der Programmunit **fLMsgHdlInit** müssen Sie die gewünschten Fehlerklassen editieren.

Maschinenfehlerklassen für anwenderdefinierte Meldungen editieren

Legen Sie die Maschinenfehlerklassen als globale Konstanten an. Sie können maximal 31 Maschinenfehlerklassen definieren. Maschinenfehlerklassen können für anwenderdefinierte Meldungen und Meldungen für Peripherie Geräte definiert werden.

Tabelle 7- 1 Globale Konstanten

```
VAR_GLOBAL CONSTANT
    // AUTOMATICALLY GENERATED CODESEQUENCE - DO NOT CHANGE!
    // <<** start label script counter**>>
    LMSGHDL_SCRIPT_COUNTER          : USINT := 6;
    // <<** end label script counter **>>
    // END OF AUTOMATICALLY GENERATED CODESEQUENCE

    //================================================================
    //           Defines for machine error classes
    // only for definition of user defined messages (application or FB
/FC)
    // user defined messages can use messageClass from 0 to 31
    //================================================================
    LMSGHDL_NO_MACHINE_ERROR_CLASS                  : SINT :=-1;
    LMSGHDL_MACHINE_ERROR_CLASS0                    : SINT :=0;
    LMSGHDL_MACHINE_ERROR_CLASS1                    : SINT :=1;
    LMSGHDL_MACHINE_ERROR_CLASS2                    : SINT :=2;
    LMSGHDL_MACHINE_ERROR_CLASS3                    : SINT :=3;
    LMSGHDL_MACHINE_ERROR_CLASS4                    : SINT :=4;
    LMSGHDL_MACHINE_ERROR_CLASS5                    : SINT :=5;
```

```
    LMSGHDL_MACHINE_ERROR_CLASS6                    : SINT :=6;
    LMSGHDL_MACHINE_ERROR_CLASS7                    : SINT :=7;
    LMSGHDL_MACHINE_ERROR_CLASS8                    : SINT :=8;
END_VAR
```

Kommentieren Sie die Initialisierung der Maschinenfehlerklassen aus und passen Sie die Maschinenfehlerklassen an die von Ihnen gewünschte Struktur an.

Tabelle 7- 2 Maschinenfehlerklassen für anwenderdefinierte Meldungen initialisieren

```
//=====================================================================
// initialize machine error classes for user defined messages
// the subindex is the eventNumber in FCLMsgHdlWriteMessageToBuffer if
// functionBlockId = 0 (no user defined message from FB / FC)
//=====================================================================
userDefinedMachineErrors[0]  := LMSGHDL_MACHINE_ERROR_CLASS1;
userDefinedMachineErrors[1]  := LMSGHDL_MACHINE_ERROR_CLASS2;
userDefinedMachineErrors[2]  := LMSGHDL_MACHINE_ERROR_CLASS3;
userDefinedMachineErrors[3]  := LMSGHDL_MACHINE_ERROR_CLASS1;
userDefinedMachineErrors[4]  := LMSGHDL_MACHINE_ERROR_CLASS1;
userDefinedMachineErrors[5]  := LMSGHDL_MACHINE_ERROR_CLASS4;
userDefinedMachineErrors[6]  := LMSGHDL_MACHINE_ERROR_CLASS5;
userDefinedMachineErrors[7]  := LMSGHDL_NO_MACHINE_ERROR_CLASS; // no ma-
chine error class used
userDefinedMachineErrors[8]  := LMSGHDL_MACHINE_ERROR_CLASS6;
userDefinedMachineErrors[9]  := LMSGHDL_MACHINE_ERROR_CLASS7;
```

Maschinenfehlerklassen für Meldungen von FBs / FCs zuordnen

Tabelle 7- 3 Maschinenfehlerklassen für Meldungen FBs / FCs initialisieren

```
//=====================================================================
// initialize machine error classes for FBs/FCs
// the subindex is the functionBlockId in FCLMsgHdlWriteMessageToBuffer
// you can use only 4 different error classes 0..3
//=====================================================================
// LMSGHDL_FB/FC1
fBFCMachineErrorClasses[0].ai8ErrorClass[0] :=
LMSGHDL_MACHINE_ERROR_CLASS0;
fBFCMachineErrorClasses[0].ai8ErrorClass[1] :=
LMSGHDL_MACHINE_ERROR_CLASS1;
fBFCMachineErrorClasses[0].ai8ErrorClass[2] :=
LMSGHDL_MACHINE_ERROR_CLASS2;
fBFCMachineErrorClasses[0].ai8ErrorClass[3] :=
LMSGHDL_MACHINE_ERROR_CLASS3;
// LMSGHDL_FB/FC2
fBFCMachineErrorClasses[1].ai8ErrorClass[0] :=
LMSGHDL_MACHINE_ERROR_CLASS2;
fBFCMachineErrorClasses[1].ai8ErrorClass[1] :=
LMSGHDL_MACHINE_ERROR_CLASS3;
fBFCMachineErrorClasses[1].ai8ErrorClass[2] :=
LMSGHDL_MACHINE_ERROR_CLASS4;
fBFCMachineErrorClasses[1].ai8ErrorClass[3] :=
```

```
LMSGHDL_MACHINE_ERROR_CLASS5;
// LMSGHDL_FB/FC3
fBFCMachineErrorClasses[2].ai8ErrorClass[0] :=
LMSGHDL_MACHINE_ERROR_CLASS5;
fBFCMachineErrorClasses[2].ai8ErrorClass[1] :=
LMSGHDL_MACHINE_ERROR_CLASS6;
fBFCMachineErrorClasses[2].ai8ErrorClass[2] :=
LMSGHDL_MACHINE_ERROR_CLASS7;
fBFCMachineErrorClasses[2].ai8ErrorClass[3] :=
LMSGHDL_MACHINE_ERROR_CLASS8;
```

Maschinenfehlerklassen für Meldungen von Peripherie Geräte zuordnen

Wenn Sie die Meldungen von Peripherie Geräten einer Maschinenfehlerklasse zuordnen möchten, müssen Sie die anlegen.

Tabelle 7- 4 Maschinenfehlerklassen für Meldungen von Peripherie Geräten initialisieren

```
//=====================================================================
// initialize message classes for peripheral devices
//=====================================================================
// SINAMICS Integrated
peripheralDevices[0].i8MachineErrorClass := LMSGHDL_MACHINE_ERROR_CLASS4;
// CU 310
peripheralDevices[1].i8MachineErrorClass := LMSGHDL_MACHINE_ERROR_CLASS6;
```

7.2 Anwenderdefinierte Meldungen editieren

In diesem Abschnitt sind zwei Beispiele für das Editieren von anwenderdefinierten Meldungen dargestellt. Sie müssen die anwenderdefinierten Meldungen in der Programmunit **fLMsgHdlInit** definieren.

Tabelle 7- 5 Beispiel einer anwenderdefinierten Meldung Nr. 4 mit 4 Beiwerten

```
// User defined message 4
// 'User defined message 4. FB-Id: /1/%d, additional value 1: /2/%d, additional value 2:
/3/%d, error code: /4/%x'
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart1  := 'User defined message 4. FB-
Id:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue1.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_FB_ID;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue1.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart2  := ', additional value 1:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue2.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_1;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue2.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart3  := ', additional value 2:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue3.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_2;
```

```
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue3.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart4  := ', error code:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue4.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_ERROR_CODE;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue4.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_HEX;
i16IndexCounter := i16IndexCounter + 1;
```

Tabelle 7- 6 Beispiel einer anwenderdefinierten Meldung Nr. 2 mit 2 Beiwerten

```
// User defined message 2
// 'User defined message 2. additional value 1:/1/%d, additional value 2:/2/%d'
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart1  := 'User defined message 2. addi-
tional value 1:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue1.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_1;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue1.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart2  := ', additional value 2:';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue2.b8ValueNumber :=
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_2;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue2.b8ValueType :=
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart3  := '';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue3.b8ValueNumber := 0;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue3.b8ValueType := 0;
userDefinedMessages[i16IndexCounter].sgLMsgHdlTextPart4  := '';
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue4.b8ValueNumber := 0;
userDefinedMessages[i16IndexCounter].ab8LMsgHdlAdditionalValue4.b8ValueType := 0;
i16IndexCounter := i16IndexCounter + 1;
```

7.3 Konstanten in der Bibliotheksunit cPublic anpassen

In diesem Anwendungsbeispiel werden Meldungen von drei FBs / FCs verwendet. Deshalb müssen Sie die Konstante in der Bibliotheksunit **cPublic** anpassen.

Tabelle 7- 7 Konstanten in **cPublic** anpassen

```
//=====================================================================
//           Defines for user messages
//=====================================================================

// if constant is TRUE AlarmS handling is active
LMSGHDL_ALARM_S_USER_MESSAGES                  : BOOL := TRUE;
// if constant is TRUE Message-Bit-Handling is active
LMSGHDL_MESSAGE_BIT_USER_MESSAGES                : BOOL := FALSE;

// max number of lines for HMI
```

```
LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI    : USINT := 10;
// max length for strings for HMI
LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI : INT   := 80;

// number of user defined application events in project (set by user)
LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS             : INT := 20;

// number of function block ids for machine error classes
LMSGHDL_NUMBER_OF_FUNCTION_BLOCK_IDS              : INT := 3;
```

Nachdem alle Anpassungen in den Units erfolgt sind, müssen Sie diese übernehmen und übersetzen. Danach müssen Sie das Projekt speichern und in das SIMOTION Gerät laden.

7.4 Funktionsaufruf

In diesem Abschnitt sind die Funktionsaufrufe für anwenderdefinierte Meldungen und anwenderdefinierte Meldungen aus FB / FC dargestellt.

Anhand der Eventnummer legen Sie die Maschinenfehlerklasse fest. In den beiden Beiwerten können Sie Zusatzinformationen übergeben lassen, welche im Text mit ausgegeben werden.

Tabelle 7- 8 Funktionsaufruf für anwenderdefinierte Meldungen einer Funktion

```
//userdefined message
fcLMsgHdlWriteUserMessageToBuffer (eventnumber := gi16EventNo
    ,additionalValue1DINT := gi32AddValue1
    ,additionalValue2DINT := gi32AddValue2);
```

Bei anwenderdefinierten Meldungen aus FB / FC können Sie maximal vier Fehlerklassen definieren. Diese werden als Parameter mit übergeben. Jeder Fehlerklasse können Sie eine Maschinenfehlerklasse zuordnen.

Tabelle 7- 9 Funktionsaufruf für anwenderdefinierte Meldungen in FB/FC

```
//userdefined message from FB / FC
fcLMsgHdlWriteFbFcMessageToBuffer (eventnumber := gi16EventNo
    ,errorClass           := gu8ErrorClass
    ,errorCode            := gb32ErrorCode
    ,functionBlockId      := gi32FBId
    ,additionalValue1DINT := gi32AddValue1
    ,additionalValue2DINT := gi32AddValue2);
```

7.5 Anzeige der Daten vom Meldungshandling im Symbolbrowser von SIMOTION SCOUT

Anzeige von Meldungen

Nachdem alle Anpassungen im SIMOTION Projekt vom Anwender erfolgt sind, werden die entsprechenden Meldungen und damit verbundene Daten im Symbolbrowser des SIMOTION SCOUT angezeigt. In diesem Abschnitt werden wichtige Daten beispielhaft dargestellt. Im Beispiel stehen vier Systemfehler und zwei anwenderdefinierte Meldungen im Projekt an.

Im Puffer der aktiven Meldungen, der Programmunit **pLMsgHdl**, werden in der Struktur *gLMsgHdlActiveMessageString* folgende Variablen angezeigt:

- Quelle der Meldung im Array *asgMessageSource*

D435.pLMsgHdl:

	Name	Data type	Status value	Display form
39	⊞ gslmsghdlactivemessagesbasedata	'slmsghdlactivemessagesbasedatatype'		
40	⊟ gslmsghdlactivemessagestring	'slmsghdlactivemessagestringtype'		
41	i16actualindex	INT	6	DEC
42	bochangesinactivestringdata	BOOL	FALSE	BOOL
43	⊞ asgmessagelevel	Array		
44	⊟ asgmessagesource	Array		
45	asgmessagesource[0]	STRING	'User defined message: '	
46	asgmessagesource[1]	STRING	'User defined message: '	
47	asgmessagesource[2]	STRING	'TO-Message: D435: Achse_rot'	
48	asgmessagesource[3]	STRING	'TO-Message: D435: Achse_blau'	
49	asgmessagesource[4]	STRING	'TO-Message: D435: Achse_blau'	
50	asgmessagesource[5]	STRING	'TO-Message: D435: Achse_rot'	
51	asgmessagesource[6]	STRING		

Bild 7-1 Quelle der Meldung

- Meldungstext im Array *asgMessageText*

D435.pLMsgHdl:

	Name	Data type	Status value
42	bochangesinactivestringdata	BOOL	FALSE
43	⊞ asgmessagelevel	Array	
44	⊞ asgmessagesource	Array	
45	⊟ asgmessagetext	Array	
46	asgmessagetext[0]	STRING[160]	'Event 2: User defined message 2. additional value 1:20, additional value 2:40'
47	asgmessagetext[1]	STRING[160]	'Event 1: User defined message 1. additional value 1:10, additional value 2:20'
48	asgmessagetext[2]	STRING[160]	'30002: Command aborted (reason: 5, command type: 00001001)'
49	asgmessagetext[3]	STRING[160]	'30002: Command aborted (reason: 5, command type: 00001001)'
50	asgmessagetext[4]	STRING[160]	'40005: Missing enable(s) (parameter1: 00000007ncorrect mode (parameter2: /2/%d)'
51	asgmessagetext[5]	STRING[160]	'40005: Missing enable(s) (parameter1: 00000007ncorrect mode (parameter2: /2/%d)'
52	asgmessagetext[6]	STRING[160]	
53	asgmessagetext[7]	STRING[160]	

Bild 7-2 Meldungstext

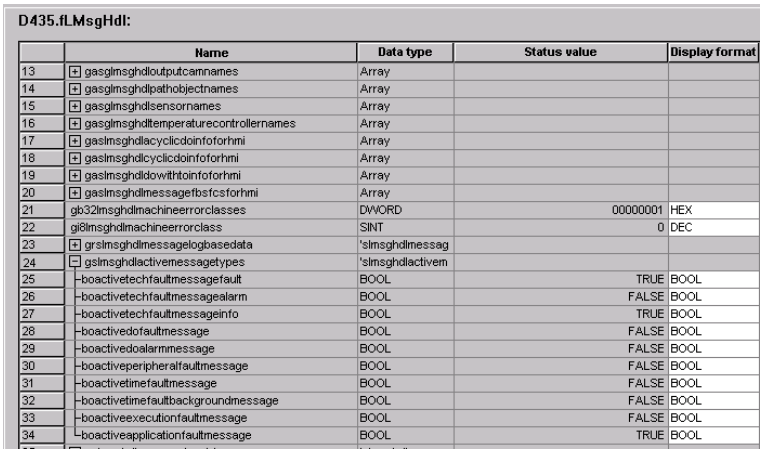
- Zeitstempel im Array *asgMessageOccured*

D435.pLMsgHdl:

	Name	Data type	Status value
42	bochangesinactivestringdata	BOOL	FALSE
43	⊞ asgmessagelevel	Array	
44	⊞ asgmessagesource	Array	
45	⊞ asgmessagetext	Array	
46	⊟ asgmessageoccured	Array	
47	asgmessageoccured[0]	STRING[23]	2010-02-24-09:06:32.631
48	asgmessageoccured[1]	STRING[23]	2010-02-24-09:06:30.990
49	asgmessageoccured[2]	STRING[23]	2010-02-24-09:06:17.721
50	asgmessageoccured[3]	STRING[23]	2010-02-24-09:06:17.721
51	asgmessageoccured[4]	STRING[23]	2010-02-24-09:06:17.712
52	asgmessageoccured[5]	STRING[23]	2010-02-24-09:06:17.712
53	asgmessageoccured[6]	STRING[23]	
54	asgmessageoccured[7]	STRING[23]	

Bild 7-3 Zeitstempel

In der Programmunit **fLMsgHdl** werden in der Struktur *gsLMsgHdlActiveMessageTypes* die Meldungstypen angezeigt.



D435.fLMsgHdl:

	Name	Data type	Status value	Display format
13	⊞ gasglmsghdloutputcamnames	Array		
14	⊞ gasglmsghdlpathobjectnames	Array		
15	⊞ gasglmsghdlsensornames	Array		
16	⊞ gasglmsghdltemperaturecontrollernames	Array		
17	⊞ gaslmsghdlacyclicdoinfoforhmi	Array		
18	⊞ gaslmsghdlcyclicdoinfoforhmi	Array		
19	⊞ gaslmsghdldowithtoinfoforhmi	Array		
20	⊞ gaslmsghdlmessagefbsfcsforhmi	Array		
21	gb32lmsghdlmachineerrorclasses	DWORD	00000001	HEX
22	gi8lmsghdlmachineerrorclass	SINT	0	DEC
23	⊞ grslmsghdlmessagelogbasedata	'slmsghdlmessag		
24	⊟ gslmsghdlactivemessagetypes	'slmsghdlactivem		
25	├boactivetechfaultmessagefault	BOOL	TRUE	BOOL
26	├boactivetechfaultmessagealarm	BOOL	FALSE	BOOL
27	├boactivetechfaultmessageinfo	BOOL	TRUE	BOOL
28	├boactivedofaultmessage	BOOL	FALSE	BOOL
29	├boactivedoalarmmessage	BOOL	FALSE	BOOL
30	├boactiveperipheralfaultmessage	BOOL	FALSE	BOOL
31	├boactivetimefaultmessage	BOOL	FALSE	BOOL
32	├boactivetimefaultbackgroundmessage	BOOL	FALSE	BOOL
33	├boactiveexecutionfaultmessage	BOOL	FALSE	BOOL
34	└boactiveapplicationfaultmessage	BOOL	TRUE	BOOL

Bild 7-4 Meldungstypen

Übersicht der globalen Variablen

A

A.1 Variablen

Globale Variablen

Im Meldungshandling sind folgende globale Variablen definiert:

Tabelle A- 1 Globale Variablen im Meldungshandling

Name	Datentyp	Unit	Verwendung
gri16LMsgHdlCounterToIniRetainBuffer	INT	pLMsgHdl	Eine Konstante wird vom Skript bei Änderungen im Meldungshandling inkrementiert. Über diese Variable wird im Hochlauf entschieden, ob die im netzaussicheren Bereich (RETAIN) gespeicherten Rohdaten initialisiert werden müssen. Retain-Daten werden gelöscht
gboLMsgHdlInitDriveReady	BOOL	pLMsgHdl	Zeigt an, dass die Initialisierungssoftware des Meldungshandlings in der BackgroundTask durchgelaufen ist. TRUE: das gesamte Meldungshandling ist aktiv FALSE: es können noch keine Meldungen übernommen werden
gboLMsgHdlActivateNewMoMaData	BOOL	pLMsgHdl	Mit TRUE werden die zur Laufzeit übergebenen Informationen für modulare Maschine aktiviert. Nach Aktivierung wird das Flag vom Meldungshandling wieder zurückgenommen.
gboLMsgHdlGlobalAcknowledge	BOOL	pLMsgHdl	Mit einer steigenden Flanke wird eine globale Quittierung aller anstehenden Fehler im Meldungshandling angestoßen. Nach der Durchführung der Quittierung wird der Wert wieder auf FALSE zurückgesetzt.
gi32LMsgHdlNumberOfMessageInLog	BOOL	pLMsgHdl	Übergabe der Nummer der zu quittierenden Meldung (Nur bei Einzelquittierung.)
gboLMsgHdlStartChangeLanguage	BOOL	pLMsgHdl	Änderung der aktiven Sprache für das Meldungshandling im Format String. Start mit steigender Flanke. Meldungshandling setzt nach der Aktion die Variable wieder auf FALSE zurück.

Name	Datentyp	Unit	Verwendung
gu8LMsgHdlActiveLanguage	USINT	pLMsgHdl	Einstellung der aktiven Sprache für das Meldungshandling. Mit TRUE, Start der Sprachumschaltung. Wird vom Meldungshandling selbstständig zurückgenommen.
gboLMsgHdlStartWriteCompleteMessageLogToStorageMedium			
	BOOL	pLMsgHdl	Mit einer steigenden Flanke wird der aktuelle MeldungsLog im Format Rohdaten und String, sowie alle notwendigen Informationen zur Interpretation der Rohdaten auf das Speichermedium des SIMOTION Gerätes geschrieben. Das Meldungshandling setzt nach der Aktion die Variable wieder auf FALSE zurück.
gu32LMsgHdlDataSetNoForExportMessageLog			
	UDINT	pLMsgHdl	Name der Datei in die das aktuelle MeldungsLog gespeichert werden soll. Default: *ds000000.dat*
gu8LMsgHdlScrollStep	USINT	pLMsgHdl	Hier kann eingestellt werden, um wie viele Meldungen die Anzeige in SIMOTION IT, bzw. im HMI nach oben, bzw. nach unten scrollen soll. Vorbelegung: LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI aus **cPublic**
gu8LMsgHdlNumberOfLinesForHMI	USINT	pLMsgHdl	Hier kann eingestellt werden, wie viele Meldungen die Anzeige in SIMOTION IT, bzw. HMI hat. Vorbelegung: LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI aus **cPublic**
gboLMsgHdlUpdateHMI	BOOL	pLMsgHdl	Aktualisieren der Anzeige auf SIMOTION IT, bzw. HMI der aktiven Meldungen. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollUp1	BOOL	pLMsgHdl	Scroll up um eine Meldung in der Liste der aktiven Meldungen. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollUp	BOOL	pLMsgHdl	Scroll up um *gu8ScrollStep* in der Liste der aktiven Meldungen. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollDown1	BOOL	pLMsgHdl	Scroll down um eine Meldung in der Liste der aktiven Meldungen.

Name	Datentyp	Unit	Verwendung
gboLMsgHdlScrollDown	BOOL	pLMsgHdl	Scroll down um *gu8ScrollStep* in der Liste der aktiven Meldungen. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlGoToTop	BOOL	pLMsgHdl	Springe an den Anfang der aktiven Meldungen.
gboLMsgHdlGoToEnd	BOOL	pLMsgHdl	Springe ans Ende der aktiven Meldungen.
gsLMsgHdlActiveMsgToHMI	sLMsgHdlHMI ActiveMsgSgType / sLMsgHdlHMI ActiveMsgBase DataType	pLMsgHdl	Liste der aktiven Meldungen, die an SIMOTION IT, bzw. HMI ausgegeben werden sollen.
gboLMsgHdlUpdateHMILog	BOOL	pLMsgHdl	Aktualisieren der Anzeige auf SIMOTION IT, bzw. HMI des MeldungsLogs. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollUp1Log	BOOL	pLMsgHdl	Scroll up um 1 in der Liste des MeldungLogs. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollUpLog	BOOL	pLMsgHdl	Scroll up um *gu8ScrollStep* in der Liste des MeldungLogs. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollDown1Log	BOOL	pLMsgHdl	Scroll down um 1 in der Liste des MeldungLogs. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlScrollDownLog	BOOL	pLMsgHdl	Scroll down um *gu8ScrollStep* in der Liste des MeldungLogs. Ausführung bei steigender Flanke. Diese wird anschließend durch das Meldungshandling wieder zurückgenommen.
gboLMsgHdlGoToTopLog	BOOL	pLMsgHdl	Springe an den Anfang der Meldungshistorie.
gboLMsgHdlGoToEndLog	BOOL	pLMsgHdl	Springe ans Ende der Meldungshistorie.
gsLMsgHdlLogMsgToHMI	sLMsgHdlHMI MsgLogSgType/ sLMsgHdlHMI ActiveMsgBase DataType	pLMsgHdl	Liste des MeldungLogs, die an SIMOTION IT, bzw. HMI ausgegeben werden sollen.

Name	Datentyp	Unit	Verwendung
gsLMsgHdlActiveMessagesBaseData	sLMsgHdlActive MessagesBase DataType	pLMsgHdl	Aktive Meldungen im Format Rohdaten.
gi16LMsgHdlNumberOfDOsInProject	INT	pLMsgHdl	Anzahl der real im Projekt vorhandenen Antriebsobjekte.
gsgLMsgHdlMessageLevel	STRING[LMSGHDL_ STRING_LENGTH_OF _MESSAGE_LEVEL]	pLMsgHdl	Hilfsvariable zur Entscheidung, ob nach einem Neustart der Maschine die Sprachdateien vom Speichermedium neu geladen werden müssen, oder nicht.
gsLMsgHdlDefaultMessages	sLMsgHdlDefault MessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten System-Meldungen
gasLMsgHdlToAxisMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_AXES_ ALARM_MESSAGES - 1] OF sLMsgHd TOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Achsen.
gasLMsgHdlToFollowingObejctsMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_ FOLLOWING_OBJECT _ALARM_MESSAGES-1] OF sLMsg HdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Gleichlaufobjekte.
gasLMsgHdlToCamsMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_CAMS_ ALARM_MESSAGES - 1] OF sLMsgH-dlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Kurvenscheiben.
gasLMsgHdlToMeasuringInputsMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_ MEASURING_INPUTS _ALARM_MESSAGES -1] OF sLMsgHdl TOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Messtaster.
gasLMsgHdlToOutputCamsMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_ OUTPUT_CAMS_ ALARM_MESSAGES - 1] OF sLMsgHdl TOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Nocken.
gasLMsgHdlToExternalEncodersMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_ EXTERNAL_ENCODE RS_ALARM_MESSAG ES - 1] OF sLMsgHdl TOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für externe Geber.
gasLMsgHdlToCamTracksMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_CAM_ TRACKS_ALARM_ME SSAGES - 1] OF sLMsgHdl TOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Nockenspuren.

Name	Datentyp	Unit	Verwendung
gasLMsgHdlToTemperatureControllers-Messages	ARRAY[0..LMSGHDL_NUMBER_OF_TEMPERATURE_CONTROLLERS_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Temperaturkanäle.
gasLMsgHdlToFixedGearsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_FIXED_GEARS_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für feste Getriebe.
gasLMsgHdlToAdditionObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_ADDITION_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Addierobjekte.
gasLMsgHdlToFormulaObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_FORMULA_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Formelobjekte.
gasLMsgHdlToSensorsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_SENSORS_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Sensoren.
gasLMsgHdlToControllerObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_CONTROLLER_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Reglerobjekte.
gasLMsgHdlToPathObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_PATH_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTOMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten TO-Meldungen für Bahnobjekte.
gai16LMsgHdlDOMessageIndex	ARRAY[1..LMSGHDL_MOST_SIGNIFICANT_DO_MESSAGE_NUMBER] OF INT	pLMsgHdl	Liste der Indizes der aktuell in der Steuerung verwendeten DO-Meldungen.

Name	Datentyp	Unit	Verwendung
gasgLMsgHdlDOMessages	ARRAY[1..LMSGHDL_NUMBER_OF_DIFFERENT_DO_MESSAGES] OF STRING[LMSGHDL_STRING_LENGTH_OF_MESSAGE_TEXT]	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten DO-Meldungen.
gsLMsgHdlSystemMessages	sLMsgHdlSystemMessagesType	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten System-Meldungen.
gasgLMsgHdlAcknowledgeClass	ARRAY OF STRING	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten Meldungen für die Quittierungsart.
gasgLMsgHdlMessageClass	ARRAY OF STRING	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten Meldungen für die Meldungsklasse.
gasLMsgHdlMessageFBsFCs	ARRAY OF STRING	pLMsgHdl	Liste der aktuell in der Steuerung verwendeten Meldungen für Meldungen durch das Meldungshandling.
gasLMsgHdlUserDefinedMessages	ARRAY[0..LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS - 1] OF sLMsgHdlUserMessagesType	fLMsgHdlInit	Liste der aktuell in der Steuerung verwendeten anwenderdefinierten Meldungen.
gsLMsgHdlActiveMessageTypes	sLMsgHdlActiveMessageTypesType	fLMsgHdl	In dieser Struktur wird angezeigt, aus welcher Meldungsquelle Meldungen aktiv sind.
gb32LMsgHdlMachineErrorClasses	DWORD	fLMsgHdl	Anzeige der aktuell aktiven Maschinenfehlerklassen.
gi8LMsgHdlMachineErrorClass	SINT	fLMsgHdl	Anzeige der aktuell höchstprioren Maschinenfehlerklasse.
gab16LMsgHdlEventflag	ARRAY[0..(LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS/16)] OF WORD	fLMsgHdl	Array zur Anzeige der aktiven anwenderdefinierten Meldungen für das Bitmeldeverfahren.
gab16LMsgHdlAckFlag	ARRAY[0..(LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS/16)] OF WORD	fLMsgHdl	Array zur Anzeige der Quittierung der anwenderdefinierten Meldungen für das Bitmeldeverfahren.
gasLMsgHdlMessageFBsFCsForHMI	ARRAY[0..LMSGHDL_NUMBER_OF_INTERNAL_APPLICATION_EVENTS - 1] OF sLMsgHdlMessagesFromMessageHandlingType	fLMsgHdl	Intern verwendete String-Texte für eigene Meldungen des Meldungshandlings. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasLMsgHdlDOWithTOInfoForHMI	ARRAY[0..LMSGHDL_NUMBER_OF_DOS_WITH_TO - 1] OF sLMsgHdlDOWithTONameType	fLMsgHdl	Namen der DOs mit TO. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.

Name	Datentyp	Unit	Verwendung
gasLMsgHdlCyclicDOInfoForHMI	ARRAY[0..LMSGHDL_NUMBER_OF_CYCLIC_DOS - 1] OF sLMsgHdlCyclicDO-NameType	fLMsgHdl	Namen der DOs mit zyklische Kommunikation. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasLMsgHdlAcyclicDOInfoForHMI	ARRAY[0..LMSGHDL_NUMBER_OF_ACYCLIC_DOS - 1] OF sLMsgHdl AcyclicDONameType	fLMsgHdl	Namen der DOs ohne zyklische Kommunikation. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlAxisNames	ARRAY[0..LMSGHDL_NUMBER_OF_AXES-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten Achsen. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlExternalEncoderNames	ARRAY[0..LMSGHDL_NUMBER_OF_EXTERNAL_ENCODERS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten externen Geber. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlMeasuringInputNames	ARRAY[0..LMSGHDL_NUMBER_OF_MEASURING_INPUTS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten Messtaster. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlOutputCamNames	ARRAY[0..LMSGHDL_NUMBER_OF_OUTPUT_CAMS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten Nocken. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlCamTrackNames	ARRAY[0..LMSGHDL_NUMBER_OF_CAM_TRACKS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten Nockenspuren. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlCamNames	ARRAY[0..LMSGHDL_NUMBER_OF_CAMS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Namen aller projektierten Kurvenscheiben. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.

Name	Datentyp	Unit	Verwendung
gasgLMsgHdlFollowingObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ FOLLOWING_OBJECT -1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Gleichlaufobjekte. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können.
gasgLMsgHdlPathObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_PATH_ OBJECT1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Bahnobjekte. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Path.
gasgLMsgHdlFixedGearNames	ARRAY[0..LMSGHDL_ NUMBER_OF_FIXED_ GEARS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten festen Getriebe. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Cam_ext.
gasgLMsgHdlAdditionObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ ADDITION_OBJECT-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Addierobjekte. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Cam_ext.
gasgLMsgHdlFormulaObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ FORMULA_OBJECT-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Formelobjekte. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Cam_ext.
gasgLMsgHdlSensorNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ SENSORS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Sensoren. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Cam_ext.
gasgLMsgHdlControllerObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ CONTROLLER_ OBJECT-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Reglerobjekte. Werden benötigt, um entsprechende Meldungen aus Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung des TP Cam_ext.
gasgLMsgHdlTemperatureController Names	ARRAY[0..LMSGHDL_ NUMBER_OF_ TEMPERATURE_ CONTROLLERS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Namen aller projektierten Temperaturkanäle. Diese werden benötigt, um entsprechende Meldungen aus dem Puffer im Format Rohdaten im HMI interpretieren zu können. Nur bei Verwendung von TControl.

Name	Datentyp	Unit	Verwendung
gsLMsgHdlMessageLogString	sLMsgHdlActive MessageStringType	fLMsgHdl	Meldungshistorie im Format String. Nur wenn Format String durch Skript angewählt wurde.
grsLMsgHdlMessageLogBaseData	sLMsgHdlMessage LogBaseDataType	fLMsgHdl	Meldungshistorie im Format Rohdaten Retain
grsLMsgHdlMessageLogBaseDataGone AndOccurred	sLMsgHdlMessage LogBaseDataGone AndOccurredType	dLMsgHdl	alternative Meldungsvariante (standardmäßig nicht aktiviert)

Interpretation der Rohdaten

B

B.1 Aufbau der Struktur

Tabelle B- 1 Tabelle zur Interpretation der Rohdaten in MeldungsLog und aktive Meldungen Teil 1

Message Source	TO-Meldungen	DO-Meldungen Fehler	DO-Meldungen Warnung	Peripherie-Meldungen
au8MessageSource [USINT]	1	2	3	4
au8MessageLevel [USINT]	Stufe [USINT]	Stufe [USINT]	Stufe [USINT]	Stufe [USINT]
au8AcknowledgeClass [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]
au8ErrorClass [USINT]				
au16Parameter1 [UINT]	TO-Type [USINT]	Achs-Referenz als Nummer [UINT]	Achs-Referenz als Nummer [UINT]	Ereignisklasse [UINT]
ai16Parameter2 [INT]	TO-Nummer [INT]	IO_Id [UINT]	IO_Id [UINT]	Störungskennung [UINT]
ab32Parameter3 [DWORD]	Meldungsnummer [DINT]	Logische Adresse DO [DINT]	Logische Adresse DO [DINT]	Logische Basisadresse INPUT [DINT]
ab32Parameter4 [DWORD]	Zusatzinfo1_DINT [DINT]	DO-Nummer [DINT]	DO-Nummer [DINT]	Logische Basisadresse OUTPUT [DINT]
ab32Parameter5 [DWORD]	Zusatzinfo2_DINT [DINT]	Fehler-Info [DINT]	Warnung-Info [DINT]	auslösender Interrupt [UDINT]
ab32Parameter6 [DWORD]	Zusatzinfo3_DINT [DINT]	Fehler-Code [UINT]	Warnung-Code [UINT]	Diagnoseadresse DP-Slave [DINT]
ab32Parameter7 [DWORD]	Zusatzinfo4_DINT [DINT]	Type of DO [INT]	Type of DO [INT]	Detailinformationen [DWORD]
ab32Parameter8 [DWORD]	Zusatzinfo5_DINT [DINT]			Master-System-Id [UDINT]
ab32Parameter9 [DWORD]				DP-Slave-Adresse [UDINT]

Message Source	TO-Meldungen	DO-Meldungen Fehler	DO-Meldungen Warnung	Peripherie-Meldungen
ab32Parameter10 [DWORD]				Slot-Nummer [UDINT]
ab32Parameter11 [DWORD]				Sub-Slot-Nummer [UDINT]
adtMessageOccured [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]
adtMessageGone [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]

Tabelle B- 2 Tabelle zur Interpretation der Rohdaten in MeldungsLog und aktive Meldungen Teil 2

Message Source	TimeFault-Meldungen	ExecutionFault-Meldungen	Meldung Neuanlauf	Meldungen FC / FB
au8MessageSource [USINT]	5	6	7	8
au8MessageLevel [USINT]	Stufe [USINT]	Stufe [USINT]	Stufe [USINT]	Stufe [USINT]
au8AcknowledgeClass [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]
au8errorClass [USINT]				au8ErrorClass [USINT]
au16Parameter1 [UINT]	taskId [UINT]	taskId [UINT]	eventSource (Hochlauf) [USINT]	au8toType [USINT]
ai16Parameter2 [INT]				ai16ToNumber [INT]
ab32Parameter3 [DWORD]	Auslösender Interrupt [UDINT]	Art des Verarbeitungsfehlers [UDINT]		eventNumber [INT]
ab32Parameter4 [DWORD]				ai32AdditionalValue1 [DINT]
ab32Parameter5 [DWORD]				ai32AdditionalValue2 [DINT]
ab32Parameter6 [DWORD]				ai32AdditionalValue3 [DINT]
ab32Parameter7 [DWORD]				ai32AdditionalValue4 [DINT]
ab32Parameter8 [DWORD]				ab32AdditionalValue5 [DWORD]
ab32Parameter9 [DWORD]				ar32AdditionalValue6 [REAL]

Message Source	TimeFault-Meldungen	ExecutionFault-Meldungen	Meldung Neuanlauf	Meldungen FC / FB
ab32Parameter10 [DWORD]				ai32FunctionBlockId [DINT]
ab32Parameter11 [DWORD]				ab32ErrorCode [DWORD]
adtMessageOccured [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]
adtMessageGone [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]

Tabelle B- 3 Tabelle zur Interpretation der Rohdaten in MeldungsLog und aktive Meldungen Teil 3

Message Source	Anwenderdefinierte Meldungen	Meldungen durch Meldungshandling	DO-Meldungen Safety
au8MessageSource [USINT]	9	10	11
au8MessageLevel [USINT]	Stufe [USINT]	Stufe [USINT]	Stufe [USINT]
au8AcknowledgeClass [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]	Quittierungsart [USINT]
au8errorClass [USINT]	u8ErrorClass [USINT]	u8ErrorClass [USINT]	u8ErrorClass [USINT]
au16Parameter1 [UINT]	eventSource [USINT]	eventSource [USINT]	Achs-Referenz als Nummer [INT]
ai16Parameter2 [INT]			IO_Id [UINT]
ab32Parameter3 [DWORD]	eventNumber [INT]	eventNumber [INT]	logischeAdresse DO [DINT]
ab32Parameter4 [DWORD]	Zusatzinfo1 [DINT]	Zusatzinfo1 [DINT]	DO-Nummer [DINT]
ab32Parameter5 [DWORD]	Zusatzinfo2 [DINT]	Zusatzinfo2 [DINT]	Safety-Info [DINT]
ab32Parameter6 [DWORD]			Safety-Code [UINT]
ab32Parameter7 [DWORD]			Safety-Code [UINT]
ab32Parameter8 [DWORD]			
ab32Parameter9 [DWORD]			
ab32Parameter10 [DWORD]		ai32FunctionBlockId [DINT]	

Message Source	Anwenderdefinierte Meldungen	Meldungen durch Meldungshandling	DO-Meldungen Safety
ab32Parameter11 [DWORD]		ab32ErrorCode [DWORD]	
adtMessageOccured [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]	Meldung gekommen [DT]
adtMessageGone [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]	Meldung gegangen [DT]

B.2 Gemeinsame Information aller Meldungen

Allgemeines

In der Struktur *grsLMsgHdlMessageLogBaseData* in der Programmunit **fLMsgHdl** liegt der globale Datenpuffer der Meldungsinformationen im Format Rohdaten. Diese Informationen werden im netzaussicheren Bereich (RETAIN) gespeichert und können wie folgt ausgewertet werden.

In *grsLMsgHdlMessageLogBaseData.i16ActualIndex* wird abgelegt, in welchem Index der letzte Eintrag im Puffer durch das Meldungshandling abgelegt wurde. Der Puffer ist ein Ringpuffer, der in aufsteigender Reihenfolge durch das Meldungshandling beschrieben wird. Wird der letzte Eintrag im Meldungspuffer gefüllt, wird der nächste Eintrag wieder an den Index 0 geschrieben. Der Puffer ist immer ausgehend vom aktuellsten Eintrag nach Zeitstempel **Meldung gekommen** sortiert.

Message source

Tabelle B- 4 Inhalt der Zellen *au8MessageSource[]*

Meldungsquelle	Wert [USINT]
unbekannte Quelle	0
TO-Meldungen	1
DO-Fehler	2
DO-Warnung	3
Peripherie-Meldungen	4
TimeFault-Meldungen	5
ExecutionFault-Meldungen	6
Meldung Neuanlauf	7
Meldungen von FBs / FCs	8
Anwenderdefinierte Meldungen	9
Meldung durch Meldungshandling	10
DO-Safety-Meldung	11

Message level

Tabelle B- 5 Inhalt der Zellen *au8MessageLevel[]*

Message level [STRING]	Wert [USINT]
unbekannt	0
Störung	1
Fehler	2
Warnung	3
Information	4
Safety-Meldung	5

Acknowledge class

Tabelle B- 6 Inhalt der Zellen *au8AcknowledgeClass[]*

AcknowledgeClass [STRING]	Wert [USINT]
unbekannt	0
keine Quittierung	1
sofort	2
Power On	3
sofort / Power on	4

Error class

Tabelle B- 7 Inhalt der Zellen *au8ErrorClass[]*

Fehlerklasse [STRING]	Wert [USINT]
Klasse0	0
Klasse1	1
Klasse2	2
Klasse3	3

Wird nur von der Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** beschrieben.

B.3 Meldungen am Technologieobjekt

TO-Meldungen

Bei Meldungen durch TOs sind folgende Zellen mit Werten belegt:

Tabelle B- 8 au16Parameter1 [UINT] = TO-Typ [USINT]

TO-Typ	Wert [USINT]
Alle Arten von Achsen	1
Gleichlaufobjekt	2
Kurvenscheiben	3
Messtaster	4
Nocken	5
Externer Geber	6
Nockenspur	7
Temperaturkanal	8
Festes Getriebe	9
Addierobjekt	10
Formelobjekt	11
Sensor	12
Reglerobjekt	13
Bahnobjekt	14

Anhand des TO-Typs kann im entsprechenden Bereich aus der Programmunit **fLMsgHdl** der zur TO-Nummer gehörende TO-Name im Format String ausgelesen werden. Die TO-Nummer entspricht dabei dem Arrayindex in dem der Name des TO abgelegt ist.

Tabelle B- 9 ai16Parameter2 [INT] = TO-Nummer [INT]

TO-Typ	Wert [USINT]	Array mit Namen des zur Nummer des TOs gehörenden TOs in der Programmunit pLMsgHdl
Alle Arten von Achsen	1	gasgLMsgHdlAxisNames
Gleichlaufobjekt	2	gasgLMsgHdlFollowingObjectNames
Kurvenscheiben	3	gasgLMsgHdlCamNames
Messtaster	4	gasgLMsgHdlMeasuringInputNames
Nocken	5	gasgLMsgHdlOutputCamNames
Externer Geber	6	gasgLMsgHdlExternalEncoderNames
Nockenspur	7	gasgLMsgHdlCamTrackNames
Temperaturkanal	8	gasgLMsgHdlTemperatureControllerNames
Festes Getriebe	9	gasgLMsgHdlFixedGearNames
Addierobjekt	10	gasgLMsgHdlAdditionObjectNames
Formelobjekt	11	gasgLMsgHdlFormulaObjectNames
Sensor	12	gasgLMsgHdlSensorNames

TO-Typ	Wert [USINT]	Array mit Namen des zur Nummer des TOs gehörenden TOs in der Programmunit pLMsgHdl
Reglerobjekt	13	gasgLMsgHdlControllerObjectNames
Bahnobjekt	14	gasgLMsgHdlPathObjectNames

Somit ist z. B. der Name des TO-Typs = 2 mit TO-Nummer = 5 in *gasgLMsgHdlFollowingObjectName[5]* im Format STRING abgelegt.

In den folgenden Variablen stehen die Nummern der zum Technologieobjekt gehörenden Meldung sowie die zur Meldung gehörenden möglichen Beiwerte. Diese Informationen werden benötigt, um den entsprechenden Meldungstext inklusive der Beiwerte selbstständig zusammenstellen zu können.

ab32Parameter3 [DWORD] = Meldungsnummer [DINT] (TSI#alarmNumber) Nummer der Technologiemeldung

ab32Parameter4 [DWORD] = Beiwert 1 [DINT] (TSI#alarmP1_DINT)

ab32Parameter5 [DWORD] = Beiwert 2 [DINT] (TSI#alarmP2_DINT)

ab32Parameter6 [DWORD] = Beiwert 3 [DINT] (TSI#alarmP3_DINT)

ab32Parameter7 [DWORD] = Beiwert 4 [DINT] (TSI#alarmP4_DINT)

ab32Parameter8 [DWORD] = Beiwert 5 [DINT] (TSI#alarmP5_DINT)

Die max. fünf Beiwerte der technologischen Meldungen, werden jeweils in den drei unterschiedlichen Datentypen DINT, UDINT und REAL durch das System automatisch bereitgestellt. Um nicht alle Beiwerte, in allen Datentypen speichern zu müssen, werden die Beiwerte als Bitmuster in einer Variablen von Datentyp DWORD abgelegt und müssen je nach gefordertem Format im Meldungstext konvertiert werden. D. h. es wird von einer technologischen Meldung gefordert, dass der Beiwert 1 im Format dezimal interpretiert werden muss, so muss der Beiwert 1 aus dem MeldungsLog von DWORD in ein dezimales Format gewandelt werden, bevor dieser in den Meldungstext integriert werden kann. Dies gilt für alle Beiwerte von technologischen Meldungen in allen geforderten, möglichen Datenformaten.

adtMessageOccured[DT] = Uhrzeit Meldung aufgetreten im Format DT

adtMessageGone[DT] = Uhrzeit Meldung gegangen im Format DT

Die übrigen Parameter sind nicht belegt.

B.4 Fehler am Antriebsobjekt

Fehler am DO

Im Meldungshandling werden drei unterschiedliche Typen von Antriebsobjekten (DO) unterschieden.

- Antriebsobjekte mit Technologieobjekt Achse
- Antriebsobjekte mit zyklischem Standardtelegramm
- Antriebsobjekte ohne zyklisches Standardtelegramm

Zur Identifikation der einzelnen DOs werden je nach Typ unterschiedliche Informationen in die Meldungspuffer eingetragen.

Die notwendigen Informationen sind in folgenden Parametern hinterlegt.

au16Parameter1 [UINT] = Achs-Referenz als Nummer [UINT] (TO-Nummer der Achse)
ai16Parameter2 [INT] = Iold [UINT] (Wert 0 = Input, 1 = Output)
ab32Parameter3 [DWORD] = logische Adresse des DO [DINT]
ab32Parameter4 [DWORD] = DO-Nummer [DINT]

Für die unterschiedlichen DO-Typen sind diese Parameter folgendermaßen belegt:

DO mit Achs-TO

au16Parameter1	= Nummer der Achse zu dem das DO gehört (siehe TO-Meldungen)
ai16Parameter2	= -1 (keine Iold)
ab32Parameter3	= -1 (keine logische Adresse übergeben)
ab32Parameter4	= 255 (keine DO-Nummer übergeben)

DO mit zyklischem Standardtelegramm

au16Parameter1	= 0 (kein TO-Achse zugeordnet)
ai16Parameter2	= 0/1 (logische Adresse ist 0 = Input oder 1 = Output)
ab32Parameter3	= logische Adresse des DO (logische Adresse aus HW Konfig)
ab32Parameter4	= 255 (keine DO-Nummer übergeben)

DO ohne zyklischem Standardtelegramm

au16Parameter1	= 0 (kein TO-Achse zugeordnet)
ai16Parameter2	= 0/1 (logische Adresse ist 0 = Input oder 1 = Output)
ab32Parameter3	= eine logische Adresse des Gerätes, auf dem sich das DO befindet
ab32Parameter4	= DO-Nummer (Antriebsobjektnummer aus den Eigenschaften des DOs)

Die Informationen, um welches DO es sich im Einzelnen handelt sind in der Programmunit **fLMsgHdl** in den Variablen *gasLMsgHdlDOWithTOInfoForHMI*, *gasLMsgHdlCyclicDOInfoForHMI* und *gasLMsgHdlAcyclicDOInfoForHMI* abgelegt.

DO mit TO-Achse

```
gasLMsgHdlDOWithTOInfoForHMI : ARRAY[0..LMSGHDL_NUMBER_OF_AXES - 1] OF
sLMsgHdlDOWithTONameType;
```

Tabelle B- 10 sLMsgHdlDOWithTONameType

Parameter	Datentyp	Beschreibung
sgDOName	STRING[25]	Name des Driveobjekts, welches mit einer Achse verbunden ist.
sgCUName	STRING[25]	Name der Control Unit auf dem sich das DO befindet.

DO mit zyklischem Standardtelegramm

```
gasLMsgHdlCyclicDOInfoForHMI : ARRAY[0..LMSGHDL_NUMBER_OF_CYCLIC_DOS - 1]
OF sLMsgHdlCyclicDONameType;
```

Tabelle B- 11 sLMsgHdlCyclicDONameType

Parameter	Datentyp	Beschreibung
i32LogAddress	DINT	Logische Adresse des Driveobjekts.
i16IoId	INT	IoId der logischen Adresse 0 = INPUT 1 = OUTPUT
sgDOName	STRING[25]	Name des Driveobjekts, welches mit einer Achse verbunden ist.
sgCUName	STRING[25]	Name der Control Unit auf dem sich das DO befindet.

DO ohne zyklischem Standardtelegramm

```
gasLMsgHdlAcyclicDOInfoForHMI : ARRAY[0..LMSGHDL_NUMBER_OF_ACYCLIC_DOS -
1] OF sLMsgHdlAcyclicDONameType;
```

Tabelle B- 12 sLMsgHdlAcyclicDONameType

Parameter	Datentyp	Beschreibung
i32LogAddress	DINT	Eine logische Adresse des Gerätes, auf dem sich das DO befindet.
i16IoId	INT	IoId der logischen Adresse 0 = INPUT 1 = OUTPUT
u8DONumber	USINT	DO-Nummer des Antriebsobjekts.

Parameter	Datentyp	Beschreibung
sgDOName	STRING[25]	Name des Driveobjekts, welches mit einer Achse verbunden ist.
sgCUName	STRING[25]	Name der Control Unit auf dem sich das DO befindet.

ab32Parameter5 [DWORD] = Beiwert zum Fehler [DINT] (Inhalt von Parameter DOx.r0949)
ab32Parameter6 [DWORD] = Fehlercode [UINT] (Inhalt von Parameter DOx.r0945)
ab32Parameter7 [DWORD] = Typ des DOs mit Fehler[INT] (Inhalt von Parameter DOx.r0107)
adtMessageOccured[DT] = Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT] = Uhrzeit Meldung gegangen im Format DT

B.5 Warnungen am Antriebsobjekt

DO-Warnungen

Die DO-Informationen bei Warnungen an Driveobjekten im Meldungshandling sind identisch zu denen der DO-Fehler aufgebaut.

ab32Parameter5 [DWORD] = Beiwert zur Warnung [DINT] (Inhalt von Parameter DOx.r2124)
ab32Parameter6 [DWORD] = Nummer der Warnung [UINT] (Inhalt von Parameter DOx.r2122)
ab32Parameter7 [DWORD] = Typ des DOs mit Warnung [INT] (Inhalt von Parameter DOx.r0107)
adtMessageOccured[DT] = Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT] = Uhrzeit Meldung gegangen im Format DT

B.6 Meldungen an der Peripherie

Peripherie-Meldungen

au16Parameter1 [UINT]	= Ereignisklasse [UINT] (TSI#eventClass)
ai16Parameter2 [INT]	= Störungskennung [UINT] (TSI#faultId)
ab32Parameter3 [DWORD]	= logische Basisadresse INPUT [DINT] (TSI#logBaseAdrIn)
ab32Parameter4 [DWORD]	= logische Basisadresse OUTPUT [DINT] (TSI#logBaseAdrOut)
ab32Parameter5 [DWORD]	= auslösender Interrupt [UDINT] (TSI#interruptId)
ab32Parameter6 [DWORD]	= Diagnoseadresse DP-Slave[DINT] (TSI#logDiagAdr)
ab32Parameter7 [DWORD]	= Detailinformationen[DWORD] (TSI#details)
ab32Parameter8 [DWORD]	= Master-System-Id der betroffenen Peripheriebaugruppe [UDINT] (wie HW Konfig)
ab32Parameter9 [DWORD]	= DP-Slave-Adresse [UDINT] (wie HW Konfig)
ab32Parameter10 [DWORD]	= Slot-Nummer [UDINT] (wie HW Konfig)
ab32Parameter11 [DWORD]	= SubSlot-Nummer [UDINT] (wie HW Konfig)
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT

B.7 TimeFault-Meldungen

TimeFault-Meldungen können lediglich in der BackgroundTask oder einer TimerInteruptTask auftreten.

Daher sind die Informationen für TimeFault-Meldungen wie folgt angelegt:

au16Parameter1 [UINT]	= TaskId [UINT]
au16Parameter1	= 1 Time-Fault in der BackgroundTask
au16Parameter1	= 2 Time-Fault in einer TimerInterruptTask
ab32Parameter3 [DWORD]	= Auslösendes Ereignis [UDINT] (TSI#interruptId)
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT

B.8 ExecutionFault-Meldungen

ExecutionFault-Meldungen werden bei Programmfehlern ausgelöst. Da nach einem Programmfehler in einer zyklischen Task das SIMOTION Gerät in den Betriebszustand STOP geht, werden diese Meldungen erst nach einem erneuten Start des SIMOTION Gerätes in das Meldungshandling übernommen. Diese aktive Meldungen und müssen quittiert werden.

Die Informationen für ExecutionFault-Meldungen sind wie folgt angelegt:

au16Parameter1 [UINT]	= TaskId [UINT] Dieser Wert wird nicht unterstützt
ab32Parameter3 [DWORD]	= Art des Verarbeitungsfehlers [UDINT] (TSI#executionFaultType)
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT

B.9 Meldungen durch Hochlauf des SIMOTION Gerätes

Die vom Meldungshandling generierte Meldung bei jedem Hochlauf des SIMOTION Gerätes hat folgende konstante Belegung:

au8MessageSource	= 7 (Meldungsquelle ist Neuanlauf)
au8MessageLevel	= 4 (Hinweis)
au8AcknowledgeClass	= 1 (keine Quittierung notwendig)
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT (Uhrzeit Neuanlauf)
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT (Uhrzeit Neuanlauf)

Die Meldungen durch einen Hochlauf des SIMOTION Gerätes werden nur in die Meldungshistorie eingetragen. Es wird keine aktive Meldung erzeugt.

B.10 Anwenderdefinierte Meldungen

Diese Meldungen werden innerhalb der Applikation mit Hilfe der Funktion **FCLMsgHdlWriteUserMessageToBuffer** durch den Anwender erzeugt.

ab32Parameter3 [DWORD]	= Nummer der anwenderdefinierten Meldung [DINT]
ab32Parameter4 [DWORD]	= Zusatzinfo1 [DINT]
ab32Parameter5 [DWORD]	= Zusatzinfo2 [DINT]
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT

B.11 Anwenderdefinierte Meldungen FB / FC und Aggregate FB

Anwenderdefinierte Meldungen bei FBs / FCs werden innerhalb der Applikation mit Hilfe der Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** erzeugt.

Die gemeinsamen Informationen einer Meldung sind wie folgt zu interpretieren:

au8ErrorClass [USINT]	= Fehlerklasse des FB/FC, welche die Maschinenfehlerklasse bestimmt, die durch den FB / FC gesetzt wird.
ab32Parameter3 [DWORD]	= Nummer der anwenderdefinierten Meldung [DINT]
ab32Parameter4 [DWORD]	= Beiwert1 [DINT]
ab32Parameter5 [DWORD]	= Beiwert2 [DINT]
ab32Parameter10 [DWORD]	= Eindeutige Nummer für den FB / FC wird durch den Anwender vergeben (functionBlockId [DINT])
ab32Parameter11 [DWORD]	= Error Code des FB/FC [DWORD]
adtMessageOccured[DT]	= Uhrzeit Meldung gekommen im Format DT
adtMessageGone[DT]	= Uhrzeit Meldung gegangen im Format DT

B.12 Meldungen durch Meldungshandling

Tritt in der Bearbeitung des Meldungshandlings ein Fehler auf, wird eine anwenderdefinierte Meldung des Meldungshandlings ausgelöst. Die Meldungen durch das Meldungshandling werden durch die Funktion **FCLMsgHdlWriteFBFCMessageToBuffer** an das Meldungshandling übergeben. Die Meldungen beginnen ab der Eventnummer 100.000 und sind fortlaufend durchnummeriert. Der Text zur Meldung mit der Nummer 100.000 steht dann im Subindex 0 usw.

Die Meldungen durch das Meldungshandling im Format String und deren Aufbau sind in der Programmunit **fLMsgHdl** im Array *gasgLMsgHdlMessageFBsFCsForHMI* hinterlegt.

Das Array ist wie folgt instanziiert:

```
gasgLMsgHdlMessageFBsFCsForHMI :
ARRAY[0..LMSGHDL_NUMBER_OF_INTERNAL_APPLICATION_EVENTS - 1] OF sLMsgH-
dlMessagesFromMessageHandlingType
```

Die Struktur *sLMsgHdlMessagesFromMessageHandlingType* hat folgenden Aufbau:

Tabelle B- 13 sLMsgHdlMessagesFromMessageHandlingType

Parameter	Datentyp	Beschreibung
sgLMsgHdlTextPart1	STRING[160]	Erster Teilstring der Meldung durch das Meldungshandling.
ab8LMsgHdlAdditionalValue1	ARRAY[0..1] OF BYTE	Angabe über Nummer und Format des möglichen ersten Beiwertes der Meldung.
sgLMsgHdlTextPart2	STRING[50]	Zweiter Teilstring der Meldung durch das Meldungshandling.

Parameter	Datentyp	Beschreibung
ab8LMsgHdlAdditionalValue2	ARRAY[0..1] OF BYTE	Angabe über Nummer und Format des möglichen zweiten Beiwertes der Meldung.
sgLMsgHdlTextPart3	STRING[50]	Dritter Teilstring der Meldung durch das Meldungshandling.
ab8LMsgHdlAdditionalValue3	ARRAY[0..1] OF BYTE	Angabe über Nummer und Format des möglichen dritten Beiwertes der Meldung.

Kontakt

C

C.1 Ansprechpartner

Siemens AG

Digital Factory

Factory Automation

Production Machines

DF FA PMA APC

Frauenauracher Straße 80

D - 91056 Erlangen

Fax.: +49 9131 98 1297

tech.team.motioncontrol@siemens.com

C.2 Internetadressen

Weitere Informationen zu den verschiedenen Themen finden Sie auf folgenden Internetseiten.

Siehe auch

SIMOTION (www.siemens.com/simotion)

SINAMICS (www.siemens.com/sinamics)

MotionControl/Applikationszentrum (www.siemens.com/motioncontrol/apc)

Verpackung (www.siemens.com/packaging)

SIMOTION Meldungshandling
(https://support.industry.siemens.com/cs/ww/de/view/48955585)

SIMATIC S7-1200/S7-1500 und SIMOTION: Azyklische Kommunikation
(https://support.industry.siemens.com/cs/ww/de/view/109479553)

Projektgenerator SIMOTION easyProject
(https://support.industry.siemens.com/cs/ww/de/view/51339107)