SIEMENS

SIMOTION
Message Handling

Application Manual

11/2016

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠DANGER
indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠WARNING
indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
GERMANY

℗ 11/2016 Subject to change

Copyright © Siemens AG 2016.
All rights reserved

Table of contents

Preface

1

1.1 General information

Note

The standard applications are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. The standard applications do not represent specific customer solutions, but are only intended to provide support for typical tasks. You are responsible for the proper operation of the described products. These standard applications do not relieve you of your responsibility regarding the safe handling when using, installing operating and maintaining the equipment. By using these standard applications, you agree that Siemens cannot be made liable for possible damage beyond the mentioned liability clause. We reserve the right to make changes and revisions to these standard applications at any time without prior notice. In the case of any differences between the suggestions made in these standard applications and other publications from Siemens, such as catalogs, the contents of the other documentation have priority.

Warranty conditions, liability, and support

If the application has been made available free of charge, the following applies:

We do not provide a warranty for any of the information contained in this document.

All other rights and claims against Siemens AG irrespective of legal basis are excluded. In particular claims for damages against Siemens AG in the case of product outage, downtime, loss of profit, either directly, indirectly or consequential damage are excluded.

This does not apply when liability is compulsory by law, e. g. in the case of the Product Liability Act, premeditation, an act of gross negligence by superiors and managerial staff of Siemens AG or in cases of fraudulent concealment of defects.

This limitation of liability also applies to sub-contractors, suppliers, delegates, superiors and managerial staff of Siemens AG.

German law shall apply to this agreement for customers with head offices in Germany; Swiss law for customers with head offices outside Germany. Application of the United Nations Convention on Contracts for the International Sale of Goods as of 11.04.1980 (CISG) is excluded.

If the application has been made available against payment, the appropriate alternative applies for the respective business transaction:

- Alternative 1: (Internal business)

If nothing else has been negotiated, then the "Conditions for the supply and services in Siemens internal business" applies in the version that is valid at the time that the equipment is purchased.

- Alternative 2: (Domestic business of Siemens AG)

If nothing else was negotiated, the "General License Conditions for Software for Automation and Drives for Customers with a Registered Office in Germany" valid at the time of sale are applicable.

- Alternative 3: (Direct export business of Siemens AG)

If nothing else has been negotiated, then the "General License Conditions for Software Products for Automation and Drives for Customers with a Seat or Registered Office outside Germany", valid at the time of sale, are applicable.

It is not permitted to distribute or duplicate these application examples in any form including excerpts thereof without the express consent of Siemens Industry Sector.

Notice regarding export identification codes

AL: N

ECCN: N

1.2 About this document

Objective

This document is intended to help the reader integrate the Message Handling application for the management of messages into the existing SIMOTION SCOUT project. The library called **LMsgHdl** provides basic functions for the display and management of messages. Previous knowledge in using the SIMOTION SCOUT engineering system is required.

Note

This document does not claim to contain all details on devices in any version or to take all conceivable operational cases and applications into account.

Should you require further information or encounter specific problems not covered in enough detail for your field of application, please contact your local Siemens office.

Target group

This document is intended for programmers, commissioning and application engineers who create applications for SIMOTION.

Restriction

In message handling, information is read from SINAMICS drives. A description of this data exchange is not part of this documentation.

Siemens Industry Online Support

This article originates from the Siemens Industry Online Support. The following link takes you directly to the download page for this document:

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

Application description 2

2.1 Field of application

2.1.1 Description

Information, faults, alarms, warnings, messages as well as user-defined messages occur in every SIMOTION application. The term **messages** is used generically in this document. Messages can occur because of external influences; e.g. changing of the status or faults on peripheral devices or drives. Messages can also be triggered in the SIMOTION motion control system. For example, system errors or technology object errors (technological alarms).

All messages are collected in a buffer (log). The user can display the current pending messages or a message log. This information can be forwarded to a higher-level controller or a control system. The message log can be sent as a file to a system specialist for a remote diagnostics, for example, when faults occur on the machine.

2.1.2 Field of application

The Message Handling application can be used universally for arbitrary applications. The CAM technology package is a minimum requirement.

2.2 Objective

2.2.1 Task

The objective of the message handling is to collect messages from various sources of the SIMOTION system and provide these to the user. A distinction is made between current pending messages and a message log.

The programmer of a SIMOTION application can further process the collected messages by either forwarding them to a higher-level system or displaying them on an HMI.

A configuration script is to be used to integrate the message handling in the SIMOTION application, by adapting programs and constants to the existing application. The message handling has a modular structure. When the configuration script is called, the user can decide which parts are to be inserted.

2.2.2 Benefits

This message handling significantly reduces the time required to create an error and message handling, which should be contained in every SIMOTION application.

The use of a configuration script for the integration in the SIMOTION application avoids the error-prone insertion of software components. The configuration script reads the project information and configures the message handling accordingly.

The message handling collects messages from the following sources:

- SIMOTION system messages
  - Technological messages of SIMOTION
  - Messages about SIMOTION system errors
  - Peripheral fault messages
- SINAMICS drive messages
  - DO safety messages
  - DO messages
- User-defined messages

The messages are stored in buffers that the users can access with their applications. These are:

- Buffer for memory-optimized data that has to be processed by the user for further use
- String-based buffers that are easy to read, but require a large amount of memory

The AlarmS handling and message bit handling can also be used.

In addition to the collection of messages, the user can also program for user-defined messages and peripheral fault messages, what effect a message has on the functioning of the machine. Depending on the severity of the message, the user can program a machine reaction.

2.3 Concept

2.3.1 Illustration of the concept

Collecting and displaying messages

The concept of collecting all messages from different sources in a uniform format was used for the message handling. In this way, all messages of the SIMOTION system can be displayed on an HMI or sent to a higher-level controller. The user can also access a uniform interface for further processing.

The following figure shows how the messages are collected from the various sources. All messages are read in by a message collector and converted to a uniform format.

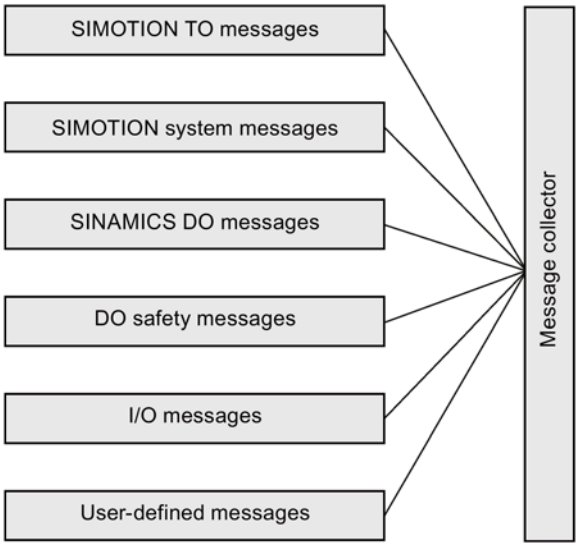

Figure 2-1 Collecting messages

Note

SIMOTION system messages include messages that are generated by timeouts in the TimeFaultTasks as well as messages generated by errors when processing a program in the ExecutionFaultTask.

The message collector writes the messages to two buffers.

- One for the current pending messages and
- One buffer with the message log

When the messages are acknowledged by the message handling, the buffer with the active messages is emptied. The messages in the message log are marked as *message gone* (time stamp when acknowledged). When the message log buffer is full, the oldest message is overwritten.

The storage format of these two buffers is called **raw data** in the following, as these two buffers are optimized for the greatest possible storage efficiency. The message information is coded with numerical values for the raw data buffers. The buffer for the message log is stored in the retentive data area (RETAIN), so that it is also available after a power failure.

Two further buffers can be integrated so that the coded numerical value of the raw data is easier to read. These contain the message information in STRING format and can be created optionally by the message handling. Corresponding to the raw data buffers, there is one STRING buffer for the active messages and one buffer for the message log. These buffers in STRING format require significantly more memory than the raw data buffers.

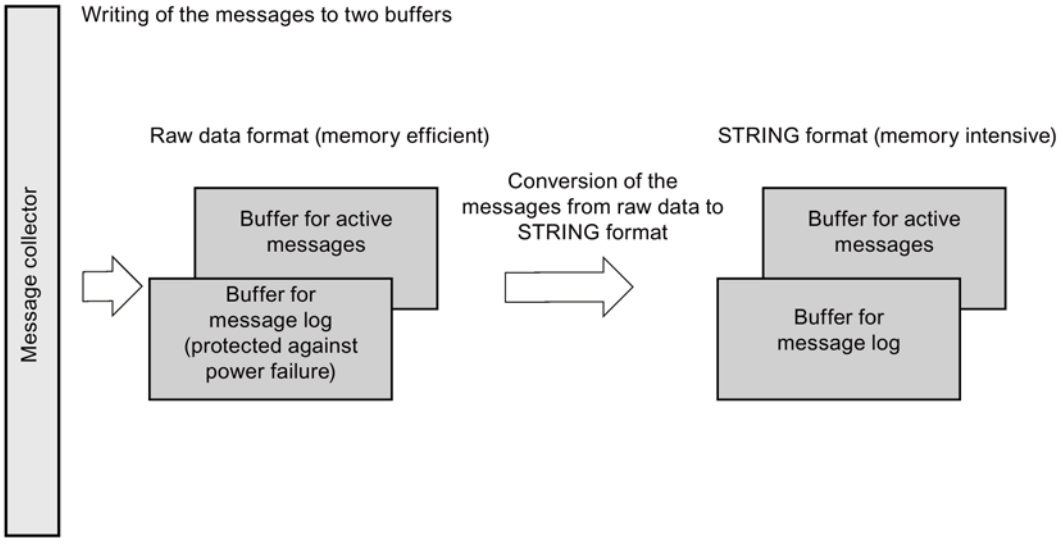

Figure 2-2 Writing messages to buffers

The AlarmS handling and the message bit handling can also be used for the display of user-defined messages on the HMI.

The user can access the buffers. Arbitrary sections of the buffers can be displayed directly on the HMI via function blocks that are also provided. When using the raw data buffers, the HMI must generate comprehensible error texts from the raw data. The texts from the STRING buffers can be used directly for the display. The buffers can also be transferred to a higher-level controller, e.g. via TCP/IP.

Remote diagnostics are facilitated when the user saves the buffer with the message log to the storage medium of the SIMOTION device when a machine fails and sends this data to a system specialist.

Note

HMI screens for the display of messages in STRING format and SIMOTION programs for the transmission to a higher-level controller are not part of the message handling. The user is responsible for this.

As the message handling has a modular structure, the user can choose whether the entire message handling or only parts thereof are to be taken into the application. The user can therefore choose from which sources the messages are to be collected, which buffers are to be used and in which way the messages are to be displayed.

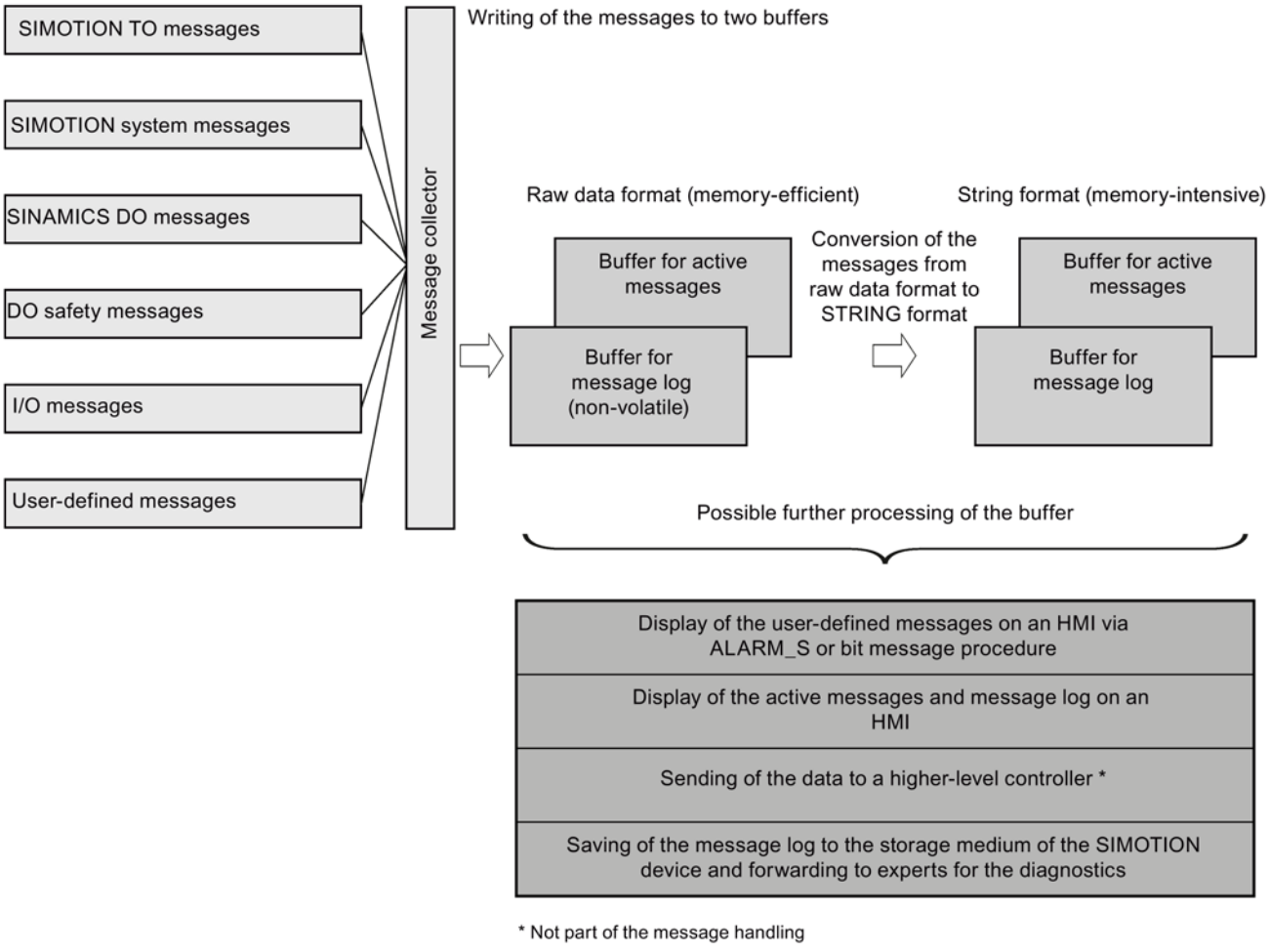

Figure 2-3 Overview of the message handling

Reaction to messages

Up to now, it has been described how messages are collected and displayed. However, most messages also require a reaction from the machine. This reaction depends on the type and source of the message. The failure of an important equipment module (machine module), e.g. when a following error occurs on the axis of an equipment module operating synchronously, requires a different reaction to that of the failure of an equipment module for filling a material storage unit. In the first case, an emergency stop of the machine would be necessary, in the second case production could continue as long as material is still available.

Note

Reactions to technological alarms and drive errors are set in the SIMOTION SCOUT engineering system and are not part of the message handling.

Each message from the user-defined messages and the peripheral fault messages can be assigned to a machine error class. These machine error classes determine the reaction of the machine when messages occur. If several messages occur simultaneously, the error

class with the highest priority determines the machine error class. For a more detailed description, see Section Defining machine error classes (Page 62).

Note

The reactions for the machine error classes must be programmed by the user. The message handling only provides a variable for the machine error class with the highest priority. All the currently active machine error classes are displayed in a variable.

2.4 System overview (example)

2.4.1 Automation overview (example)

The message handling collects messages from the peripheral devices connected to the SIMOTION device. The following figure shows an automation solution with a SIMOTION D device. Messages from the peripheral devices (an ET200M, an ET200S and a SINAMICS S120 CU320 in the figure) connected via PROFIBUS or PROFINET are forwarded to the message handling.

With SINAMICS drives, the messages from drive objects (DOs) can also be accessed. The SINAMICS drive can be connected via the integrated PROFIBUS (for SIMOTION D, this is the SINAMICS Integrated) and also via PROFIBUS or PROFINET. Data exchange is performed via acyclic services. In this way, it is possible to perform a detailed search for errors in SINAMICS drives.

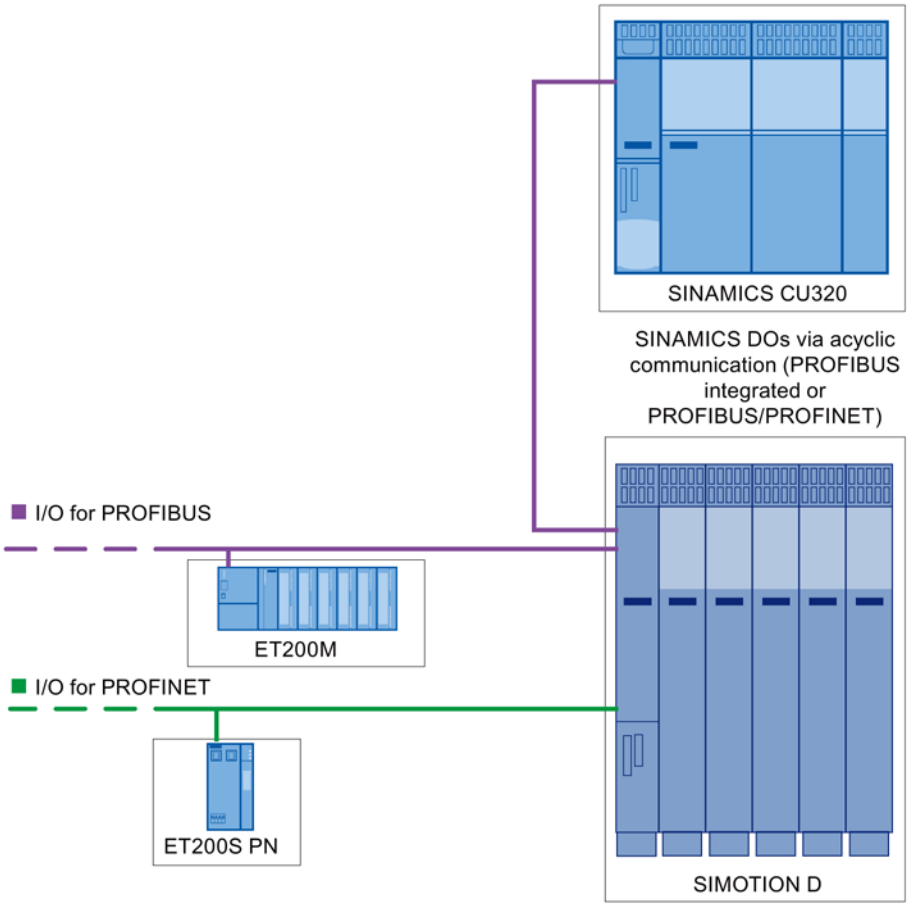

Figure 2-4 Example of an automation solution

2.4.2 Hardware structure

The message handling has been created for the SIMOTION motion control system and requires a controller of this type. It can be used for all versions of SIMOTION devices (SIMOTION D, C and P). In principle, the drive type is irrelevant when using a PROFIdrive standard telegram for a SINAMICS S120 drive, however, the status word of the drive is addressed directly, which may not function when the interface has been defined differently.

2.4.3 System requirements

The message handling has been created and tested for the software version as of SIMOTION SCOUT V4.1 SP4 with SINAMICS 2.5 and 2.6. The CAM technology package is a minimum requirement.

2.4.4 Scope of delivery

You will find the following data on the supplied medium:

- The LDPV1 and LMsgHdl libraries in XML format
- The program units of the message handling in XML format
- SIMOTION IT pages
- Files in XML format for the language selection of the message texts in German, English, French and Italian.

Application structure

3

3.1 Structure of the libraries

3.1.1 Overview of the libraries

The following libraries are used for the message handling:

- **LDPV1** library for the acyclic data exchange with SINAMICS drives.
  The descriptions of the blocks and functionalities of this library can be found on the Utilities & Applications storage medium, which is part of SIMOTION SCOUT.
- **LMsgHdl** library for the functionalities of the message handling. The library is split into various units. The LMsgHdl library has been created for the message handling and is described in this document.

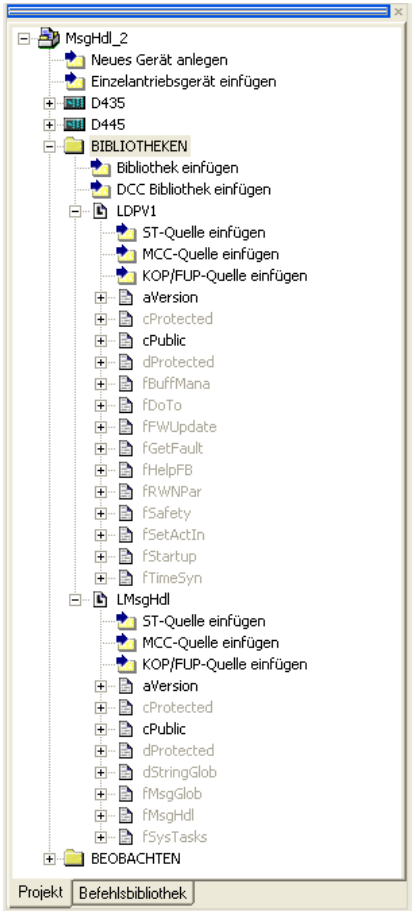

Figure 3-1 Overview of the libraries for the message handling

3.1.2 Structure of the LMsgHdl library

The following table lists the units of the **LMsgHdl** library. The user has access to two units of the library, all other units have know-how protection. Only the units without know-how protection are described.

The **aVersion** unit is used to identify the version history of the library. No source code is contained in this unit.

The constants of the **cPublic** unit are described in Section Constants (Page 22).

Table 3- 1 Structure of the LMsgHdl library

Unit name	Use	Know-how protection
aVersion	Unit of the version overview, change list	No
cProtected	Unit of the definition of the protected constants	Yes
cPublic	Unit of the definition of the constants that can be changed by the user	No
dProtected	Unit of the protected data	Yes
dStringGlob	Unit for texts of the string-based buffers in German, English, French and Italian.	Yes
fMsgGlob	Unit for functions of the string-based buffers	Yes
fMsgHdl	Unit for functions of the message handling	Yes
fSysTasks	Unit for functions to read out messages from SIMOTION fault tasks	Yes

3.2 Structure of the units in the SIMOTION project

Units of the message handling

Units are created in the SIMOTION application when the configuration script of the message handling is executed. The interfaces for the operation of the message handling as well as the message buffers are defined in global variables. The following units are available in the application of the message handling:

Table 3- 2 Units of the message handling in the SIMOTION application

Unit	Use	Know-how protection
fLMsgHdlInit	Functions that have to be adapted by the configuration script or the user in the specific project	No
fLMsgHdl	Functions for the message handling	Yes
pLMsgHdl	Unit for programs of the message handling	Yes
dLMsgHdl	Alternative message variant (not activated by default)	No

Note

The global variables declared in the program units can be used in the user application and monitored via the symbol browser. A description of the global variables can be found in Overview of the global variables (Page 107).

Programs in the message handling

The **pLMsgHdl** unit contains programs that are assigned to the execution system by the configuration script. These are used for the initialization, to collect messages and to process messages in the buffers. The unit has know-how protection and the user cannot make any changes.

Table 3- 3 Programs in the pLMsgHdl unit

Name of the program	Task level	Use
pLMsgHdlStartupMessageHandling	StartupTask	• Initialization of the data • Assignment of the TO references, the DO addresses and the I/O addresses • Setting the machine error classes
pLMsgHdlTechnologicalMessage	TechnologicalFaultTask	Reading in technological messages
pLMsgHdlPeripheralMessage	PeripheralFaultTask	Reading in peripheral messages
pLMsgHdlTimeFaultMessage	TimeFaultTask	Reading in timeout messages
pLMsgHdlTimeFaultBackgroundMessage	TimeFaultBackgroundTask	Reading in timeout messages in the BackgroundTask
pLMsgHdlExecutionFaultMessage	ExecutionFaultTask	Reading in messages when executing the SystemFaultTask
pLMsgHdlMain	BackgroundTask	Call of all message handling programs in the BackgroundTask

3.3 Constants

3.3.1 Public constants

The following constants are preset by the configuration script and should not be changed by the user.

Note

The constants in the **cPublic** unit for the number of TOs and DOs in the project are written by the configuration script to their correct values. These constants may not be changed by the user. If the project configuration changes, the configuration script must be called again. The message log in the retentive data area (RETAIN) is then reinitialized when the SIMOTION device is restarted.

Table 3- 4 Public constants in the **cPublic** unit of the **LMsgHdl** library (preset by the configuration script)

Name	Value	Use
LMSGHDL_LENGTH_OF_MESSAGE_LOG	200	Length of the buffer for the message log (applies for raw data and STRING buffers)
LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES	100	Length of the buffer for active message (applies for raw data and STRING buffers)
LMSGHDL_LANGUAGE_FOR_MESSAGE_STRING	9	Setting of the language (default setting 9 = English) (STEP 7 notation)
LMSGHDL_NUMBER_OF_AXES	1	Number of axes in the project (real and virtual)
LMSGHDL_NUMBER_OF_EXTERNAL_ENCODERS	1	Number of external encoders in the project
LMSGHDL_NUMBER_OF_MEASURING_INPUTS	1	Number of measuring inputs in the project
LMSGHDL_NUMBER_OF_OUTPUT_CAMS	1	Number of output cams in the project
LMSGHDL_NUMBER_OF_CAM_TRACKS	1	Number of cam tracks in the project
LMSGHDL_NUMBER_OF_CAMS	1	Number of cams in the project
LMSGHDL_NUMBER_OF_FOLLOWING_OBJECTS	1	Number of following objects in the project
LMSGHDL_NUMBER_OF_PATH_OBJECTS	1	Number of path objects in the project
LMSGHDL_NUMBER_OF_FIXED_GEARS	1	Number of fixed gears in the project
LMSGHDL_NUMBER_OF_ADDITION_OBJECTS	1	Number of addition objects in the project
LMSGHDL_NUMBER_OF_FORMULA_OBJECTS	1	Number of formula objects in the project
LMSGHDL_NUMBER_OF_SENSORS	1	Number of sensor objects in the project
LMSGHDL_NUMBER_OF_CONTROLLER_OBJECTS	1	Number of controller objects in the project
LMSGHDL_NUMBER_OF_TEMPERATURE_CONTROLLERS	1	Number of temperature controllers in the project
LMSGHDL_NUMBER_OF_TOS_WITH_DO	1	Number of DOs with technology object (electric axes)

Name	Value	Use
LMSGHDL_NUMBER_OF_CYCLIC_DOS	1	Number of DOs with cyclic data exchange (without DOs that are connected to TO axes)
LMSGHDL_NUMBER_OF_ACYCLIC_DOS	1	Number of DOs with acyclic data exchange (no configured telegram)
LMSGHDL_NUMBER_OF_PERIPHERAL_DEVICES	1	Number of peripheral devices
LMSGHDL_MAX_NUMBER_OF_SYSTEM_TASKS	53	Number of existing tasks in the execution system of the SIMOTION project

Non-editable constants that can be used by the user

The following constants can be used by the user to edit the user-defined message texts. These constants may **not** be changed by the user.

Table 3- 5 Public constants in the **cPublic** unit of the **LMsgHdl** library (that can be used by the user)

Name	Value	Use
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_1	1	Can be used to transfer additional value 1 (additionalValue1) for a user-defined message.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_2	2	Can be used to transfer additional value 2 (additionalValue1) for a user-defined message.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_FB_ID	3	Can be used to transfer additional value FB-ID (functionBlockId) for a user-defined message.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_ERROR_CODE	4	Can be used to transfer additional value error code (errorCode) for a user-defined message.
LMSGHDL_USER_MESSAGE_ADDITIONAL_VALUE_REAL	5	Can be used to transfer additional value REAL (additionalValueReal) for a user-defined message.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_DINT	0	Can be used to transfer data type DINT for the output of a user-defined message.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_HEX	1	Can be used to transfer data type HEX for the output of a user-defined message.
LMSGHDL_USER_MESSAGE_VALUE_TYPE_REAL	2	Can be used to transfer data type REAL for the output of an additional value of a user-defined message.

3.3.2 Changeable public constants

The following constants are not preset by the configuration script and must be changed by the user as required.

Note

The value of the constants must be at least 1 (one). The value 0 (zero) is not permitted.

Table 3- 6 Changeable public constants in the **cPublic** unit of the **LMsgHdl** library

Name	Value	Use
LMSGHDL_NUMBER_OF_STRING_MESSAGES_PER_CYCLE_IN_STARTUP		
	3	Number of messages in STRING format that are generated in the initialization during startup. The startup time of the message handling can be reduced via this constant. If the value selected is too large however, this can result in a timeout in the BackgroundTask.
LMSGHDL_MAX_NUMBER_OF_NEW_MESSAGES_PER_CYCLE		
	1	Number of new messages that can be taken into the message buffer in one background cycle. The default value should be retained here, if possible.
LMSGHDL_AUTO_SAVE_MESSAGE_BUFFER_TO_STORAGE_MEDIUM		
	FALSE	Activation/deactivation of the automatic saving of the message buffer to the storage medium of the SIMOTION device function
LMSGHDL_MAX_NUMBER_OF_DATASETS_ON_STORAGE_MEDIUM		
	5	Number of files that are created with AutoSave = TRUE
LMSGHDL_NUMBER_OF_EXECUTION_FAULT_MESSAGES		
	2	Size of the buffer to collect messages for program execution errors during runtime. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_TECH_FAULT_MESSAGES		
	100	Size of the buffer to collect the technology messages. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_PERIPHERAL_FAULT_MESSAGES		
	50	Size of the buffer to collect the peripheral fault messages. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_TIME_FAULT_MESSAGES		
	5	Size of the buffer to collect the timeout messages. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_APPLICATION_MESSAGES		
	20	Size of the buffer to collect the user-defined messages within a task. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_DO_FAULT_MESSAGES		
	50	Size of the buffer to collect the fault messages on drive objects. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_DO_ALARM_MESSAGES		
	50	Size of the buffer to collect the alarm messages on drive objects. The buffer can be enlarged if there is an overflow.
LMSGHDL_NUMBER_OF_DO_SAFETY_MESSAGES		

Name	Value	Use
	50	Size of the buffer to collect the safety messages on drive objects. The buffer can be enlarged if there is an overflow.
LMSGHDL_ALARM_S_USER_MESSAGES		
	FALSE	TRUE: Use of AlarmS for the message display on the HMI. FALSE: No use of AlarmS.
LMSGHDL_MESSAGE_BIT_USER_MESSAGES		
	FALSE	TRUE: Use of message bit handling for the message display on the HMI. FALSE: No use of the message bit handling.
LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI		
	10	Maximum number of lines for the display of a message buffer (actual number is transferred separately for the message log and display of the active errors).
LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI		
	80	Maximum number of characters for message texts in the STRING format that can be transferred via the internal function blocks to an HMI or SIMOTION IT.
LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS		
	10	Number of user-defined messages.
LMSGHDL_NUMBER_OF_FUNCTION_BLOCK_IDS		
	1	Number of user-defined messages through FBs/FCs. These FB/FC messages are part of the total number of messages in the LMSGHDL_MAX_NUMBER_OF_USER_DEFINED_EVENTS constant.
LMSGHDL_MACHINE_ERROR_CLASS_ERROR_IN_MESSAGEHANDLING		
	0	Specifies the message class that sets the messages, which have been issued by the message handling itself, in the message handling.

3.4 Core functions and components

3.4.1 Overview of the core functions and required components of the message handling

The message handling has the following core functions:

- Managing and displaying messages
- Using the AlarmS handling or message bit handling
- Acknowledging active messages
- Saving current messages to the storage medium of the SIMOTION device
- Setting the language of the message texts

Other components required in the message handling:

- Buffer management for acyclic DP-V1 data exchange services
- Startup check of the SIMOTION device

3.4.2 Description of the core functions and required components

3.4.2.1 Buffer management

Buffer management for acyclic DP-V1 data exchange services

The message handling uses the acyclic DPV1 data exchange service to determine information on SINAMICS modules. To avoid a collision of the individual data exchange jobs, the message handling uses the global buffer management (*pGlobalBufferManager* program) of the **LDPV1** library. This procedure is required as only one acyclic data exchange job can be processed simultaneously for each SINAMICS device.

Therefore, when using the message handling it is essential to check that in the entire application all further acyclic data exchange jobs are also issued via the global buffer management of the LDPV1 library. The message handling uses the buffer with the identifier zero (0) for all drive units, i.e. no other buffers may be used in applications in which acyclic data exchange is used, in order to coordinate accesses. For more detailed information, refer to the documentation on the **LDPV1** library.

The functions and function blocks of the **LDPV1** library should be used instead of system commands for the data exchange in the drive in order to avoid collisions in the data exchange. These blocks are already configured for the use of the buffer management.

Note

You must ensure that the *pGlobalBufferManager* program is available in the project and assigned to the BackgroundTask. The program is supplied with the LDPV1 library.

3.4.2.2 Description of the buffers

General

Four buffers are created in the message handling. Depending on the setting in the configuration script, either only the buffers for the raw data or all four buffers are supplied with data:

- Buffer for message log as raw data (message history)
- Buffer for active messages, raw data
- Buffer for message log as strings (message history, strings), optional
- Buffer for active messages, strings, optional

These buffers are described in the following sections. For the interpretation of messages in raw data format, see Section Interpretation of the raw data (Page 115).

Note

The message log is saved as soon as the buffer is full with the *AutoSave* function. All messages that do not have a *gone* time stamp at this time are saved without this time stamp.

Buffer for message log as raw data (message history)

All the messages that have occurred are displayed in the form of raw data in the buffer for the entire message log. This data is stored as global data in the retentive area (RETAIN):

- Data of the buffer for active messages
- Time stamp **Message gone**

If the memory requirement of the message log is greater than the RETAIN data area, the data area for the SIMOTION device can be switched from RETAIN to NON-RETAIN. To do this, the preprocessor definition LMSGHDL_NO_RETAIN_BUFFER must be set in the fLMsgHdl program unit, see Suppressing messages (Page 55).

As the message-specific information differs greatly depending on the source, this information is collected in a generally defined structure. The result is that not all elements of this structure are always filled.

To be able to evaluate this message information, it must be known how the individual messages from the various sources are to be interpreted.

As the buffer for the message log can be transferred to an HMI, the structure in STRUCT OF ARRAY is used for the buffer. A higher performance is achieved in this way. This buffer is in the **fLMsgHdl** program unit and is called *grsLMsgHdlMessageLogBaseData*.

- 64 bytes are required per message in the buffer for the entire message log.
- The length of the storage area for the entire message log can be set via the LMSGHDL_LENGTH_OF_MESSAGE_LOG constant. The default setting is for 200 entries.
  The entire message log therefore requires approx. 12 KB memory in the retentive data area (RETAIN).
  The SIMOTION D410 is an exception, the default setting is for 150 entries.

The buffer for the message log as raw data is designed as a ring buffer.

Buffer for active messages, raw data

All the messages that have not been acknowledged or cannot be acknowledged are displayed in the buffer for active messages. This data is stored as global data.

The following information is stored for each message:

- Identifier for the message source
- Level of the message (error, fault, alarm)
- Type of acknowledgement for the message, e.g. some DO messages require Power On as acknowledgement

- Message class
- Message information as raw data, as it occurs in the application
- Time stamp **when message occurred**

The structure and use of the data is identical to that of the message log in raw data format. Only the time stamp **Message gone** is missing.

This buffer is in the **pLMsgHdl** program unit and is called *gsLMsgHdlActiveMessagesBaseData*.

- 51 bytes are required per message in the buffer for the active messages.
- The length of the storage area for the active messages can be set via the LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES constant. The default setting is for 100 entries.
  The entire message log for active messages as raw data therefore requires approx. 5.1 KB memory in the global data area.

The buffer for active messages is not a ring buffer and is written successively with the entries found in the message log. If more messages are active than the length of the buffer, the remaining active messages are not displayed in the list. A separate message is displayed for this.

Buffer for message log as strings (message history, strings)

All the messages that have occurred are displayed in STRING format in the buffer for the message log. This data is stored as global data. Its use is optional and can be selected or deselected during the configuration.

The string texts are stored either in German, English, French or Italian on the controller. Other languages (only ASCII character code) are possible and can be loaded from the storage medium of the SIMOTION device to the system. For more detailed information, see Section Loading the language from the storage medium of the SIMOTION device (Page 41).

The following information is stored for each message:

- Information of the active message
- Time stamp **Message gone**

This buffer is in the **fLMsgHdl** program unit and is called *gsLMsgHdlMessageLogString*.

- 314 bytes are required per message in the buffer for the entire string message log.
- The length of the storage area for the entire string message log can be set via the LMSGHDL_LENGTH_OF_MESSAGE_LOG constant. The default setting is for 200 entries.
  The entire string message log therefore requires approx. 62.8 KB memory in the global data area.
  The SIMOTION D410 is an exception, the default setting is for 150 entries.

The buffer for the message log as raw data is designed as a ring buffer.

Buffer for active messages, strings

All the messages that have not been acknowledged or cannot be acknowledged are displayed in the buffer for active messages. This data is stored as global data. Its use is optional and can be selected or deselected during the configuration.

The string texts are available either in German, English, French or Italian. Other languages (only ASCII character code) are possible and can be loaded from the storage medium of the SIMOTION device to the system. For more detailed information, see Section Loading the language from the storage medium of the SIMOTION device (Page 41).

The following information is stored for each message:

- Identifier for the message source
- Level of the message (error, fault, alarm)
- Type of acknowledgement for the message, e.g. some DO messages require Power On as acknowledgement
- Language-dependent message text including additional information
  This achieves that all messages from all sources have the same form of message despite different information.
- Category of message
- Time stamp **when message occurred**

This buffer is in the **pLMsgHdl** program unit and is called *gsLMsgHdlActiveMessageString*.

- 291 bytes are required per message in the buffer for the active messages.
- The length of the storage area for the active messages can be set via the LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES constant. The default setting is for 100 entries.
  The entire message log for active messages as raw data therefore requires approx. 29.1 KB memory in the global data area.

The buffer for active messages is not a ring buffer and is written successively with the entries found in the message log. If more messages are active than the length of the buffer, the remaining active messages are not displayed in the list. A relevant message is also not generated.

3.4.2.3 Functions for entering user-defined messages

The **FCLMsgHdlWriteUserMessageToBuffer** and **FCLMsgHdlWriteFBFCMessageToBuffer** functions are used to transfer the user-defined messages to the message handling and take over the associated message class. With the aid of these functions, the corresponding message is also issued in the system when the AlarmS handling or message bit handling is selected. The functions can be called in all tasks of the execution system.

The AlarmS messages and also the appropriate bits for the message bit handling, are set by these functions.

The texts for the user-defined messages must be created by the user in all the required languages. The functioning and handling of the various languages is the same as for the system messages.

Note

Each user-defined message can only be active once in the system. If a message is active and transferred again to the message handling, it is not entered. Only after a user-defined message has been acknowledged, can it be issued again.

There are two different types of user-defined messages:

- User-defined messages of the application
- User-defined messages of functions and function blocks

See also

FCLMsgHdlWriteUserMessageToBuffer and FCLMsgHdlWriteFBFCMessageToBuffer functions (Page 85)

3.4.2.4 AlarmS

AlarmS is a built-in system function and can be used by the Message Handling application. To do this, the user activates the AlarmS handling in the SIMOTION project. When AlarmS is activated, the appropriately configured AlarmS message is issued for each user-defined message. For more detailed information, see Section Embedding of the AlarmS handling or message bit handling (Page 59).

3.4.2.5 Message bit handling

The message bit handling is an available function and can be used by the Message Handling application. To do this, the user activates the message bit handling in the SIMOTION project. For more detailed information, see Section Embedding of the AlarmS handling or message bit handling (Page 59).

3.4.2.6 Response to execution faults in programs

The response to program faults is set by the configuration script for all tasks on the **ExecutionFaultTask**. If a program fault occurs in a sequential task, e.g. access outside of the array limits, the ExecutionFaultTask is started. The task, in which this fault has occurred, is aborted, the SIMOTION device remains in RUN mode. When the ExecutionFaultTask is run through, the message information that is output is stored in a buffer of the message handling. This buffer is in the retentive data area (RETAIN). As the SIMOTION device then automatically goes into STOP mode, this message can no longer be entered in the message buffer. If the SIMOTION device is set again to RUN mode, the message is taken into the message buffer and appears in the message log and the active messages. This message can be acknowledged in the message handling.

3.4.2.7 Message handling startup

The following actions are performed in the Message Handling library when the message handling starts up:

- Initialization of the message buffer for the message log in raw data format. All messages that were active when the machine was shut down are automatically acknowledged during startup.
- After the configuration script is run through, the message log in raw data format is always deleted in the retentive data area (RETAIN). If changes are made in the project, the configuration script must be run through again. After the configuration script has been completed, it is not certain that the configured information determined by the configuration script matches the information already available in the raw data message log.
- If available on the storage medium of the SIMOTION device, loading of the language stored in the message handling. This action is only performed when the STRING format has been selected in the message handling.
- Generation of a new message for the restart of the SIMOTION device.
- Monitoring of the startup of all the devices configured on the SIMOTION device. If a configured device is missing or not ready, a message is entered in the message handling.
- Determination of special information about all the configured drive objects on SINAMICS modules.
- Reading out of the names of drive objects for the message handling in STRING format.
- Time synchronization of all SINAMICS modules to RTC (Real Time Clock) of SIMOTION (only with selected time synchronization and when using standard telegram 39x on the control unit of a SINAMICS object).
- The message buffers are generated when the message log in STRING format has been selected in the message handling.
- If an error occurs in the buffer management during startup of the message handling, the startup may never be completed (variable *boInitDriveReady* = TRUE). For this reason, the startup check is immediately terminated when the buffer management signals an error. This error is transferred to the message handling and can therefore be output.

Note

If the buffer management in the STRING format is used with a large buffer for the message log, the generation of the STRING buffer during startup can take a long time. To accelerate this process, the LMSGHDL_NUMBER_OF_STRING_MESSAGES_PER_CYCLE_IN_STARTUP constant has been created in the **cPublic** unit of the **LMsgHdl** library. Depending on its value, several strings are copied in succession in each cycle in the BackgroundTask. This significantly reduces the startup time.

No old messages may be active after a restart of the SIMOTION device. All messages that were not gone before startup are automatically acknowledged during startup. After all messages have been acknowledged, a new message for the startup of the SIMOTION device is entered in the message log. This entry receives the time stamp **time occurred = time gone = current value of the RTC**.

3.4.2.8 Acknowledgement of the active messages

The global acknowledgement of all messages is performed via the global variable *gboLMsgHdlMsgHdlGlobalAcknowledge* in the **pLMsgHdl** program unit. The variable must be set to TRUE by the application. The rising edge triggers a global acknowledgement of all active messages in the message handling. After acknowledgement, the variable is automatically reset to FALSE. Acknowledgement by the message handling acknowledges all the active faults and messages on SIMOTION and SINAMICS. If the AlarmS handling or message bit handling is active, these active messages are also acknowledged in the system. All active messages (except for alarms on drive objects) are reset in the message handling. If faults and messages are still present after the acknowledgement, these are taken into the message handling again.

Note

All faults and messages that occur in the controller or are displayed in the message handling, may only be acknowledged via the message handling. If an acknowledgement is triggered that bypasses the message handling, this cannot be detected by the message handling. The display in the message handling would then be incorrect.

3.4.2.9 Filtering messages to an HMI / SIMOTION IT

General

The filters are used on the output interface to an HMI / SIMOTION IT. For this purpose, the implementation within the **FBLMsgHdlActiveMsgBaseDataToHMI**, **FBLMsgHdlMsgLogBaseDataToHMI**, **FBLMsgHdlActiveMsgSgToHMI** and **FBLMsgHdlMsgLogSgToHMI** function blocks is executed. All messages are entered in the respective global buffer. This means that within the output on HMI / SIMOTION IT it is possible to select which message sources are to be displayed and which are not.

Message sources

A distinction is made between the following message sources:

- Messages from technology objects
  If selected, all messages (active or message log) that have been generated by technology objects would be output to an HMI or SIMOTION IT.
- DO alarm
  If selected, all alarms (active or message log) that have been generated by drive objects would be output to an HMI or SIMOTION IT.
- DO warnings
  If selected, all warnings (active or message log) that have been generated by drive objects would be output to an HMI or SIMOTION IT.
- DO safety messages
  If selected, all safety messages (active or message log) that have been generated by drive objects would be output to an HMI or SIMOTION IT.

- Messages from I/O modules
  If selected, all messages (active or message log) that have been generated by I/O modules would be output to an HMI or SIMOTION IT.
- User-defined messages
  If selected, all messages (active or message log) that have been generated by the user would be output to an HMI or SIMOTION IT.
- System messages
  If selected, all messages (active or message log) that have been generated by the system would be output to an HMI or SIMOTION IT. This includes TimeFault messages, TimeFault messages of the BackgroundTask and ExecutionFault messages.
- Messages by message handling
  If selected, all messages (active or message log) that have been generated by the message handling would be output to an HMI or SIMOTION IT.

If one of the sources described here is deselected, the message is still entered in the corresponding buffer, but does not appear at the output of the function blocks described above for output to an HMI or SIMOTION IT. These messages are only filtered out for the output.

The filters are selected in the *gsLMsgHdlFilterToHMI* variable of the **pLMsgHdl** program unit. This variable transfers the filter criteria to all output function blocks for an HMI or SIMOTION IT.

Structure of the gsLMsgHdlFilterToHMI variable

The *gsLMsgHdlFilterToHMI* variable is of the *sLMsgHdlFilterToHMIType* type and has the following structure. The *gsLMsgHdlFilterToHMI* variable can be transferred to the already existing HMI **FBLMsgHdlActiveMsgSgToHMI**, **FBLMsgHdlMsgLogSgToHMI**, **FBLMsgHdlActiveMsgBaseDataToHMI** and **FBLMsgHdlMsgLogBaseDataToHMI** function blocks. The outputs resulting there are taken into account with the relevant filter information.

Table 3- 7 Structure of sLMsgHdlFilterToHMIType

Parameter	Data type	Initial value	Description
boShowTOMessages	BOOL	TRUE	With TRUE, all messages of all TOs are displayed, with FALSE, they are filtered out.
boShowDOWarnings	BOOL	TRUE	With TRUE, all warnings of all DOs are displayed, with FALSE, they are filtered out.
boShowDOAlarms	BOOL	TRUE	With TRUE, all alarms of all DOs are displayed, with FALSE, they are filtered out.
boShowDOSafetyMessages	BOOL	TRUE	With TRUE, all safety messages of all DOs are displayed, with FALSE, they are filtered out.
boShowPeripheralMessages	BOOL	TRUE	With TRUE, all messages of all I/O modules are displayed, with FALSE, they are filtered out.

Parameter	Data type	Initial value	Description
boShowSystemMessages	BOOL	TRUE	With TRUE, all system messages are displayed, with FALSE, they are filtered out.
boShowUserDefinedMessages	BOOL	TRUE	With TRUE, all user-defined messages are displayed, with FALSE, they are filtered out.
boShowMessagesFromMsgHdl	BOOL	TRUE	With TRUE, all messages generated by the message handling are displayed, with FALSE, they are filtered out.

3.4.2.10 Modular machine

General

With the message handling, it is possible to suppress message entries in the message buffer for specific objects.

Example 1

The motor of a conveyor belt is defective. If the TO or DO fails, errors of this axis or drive are entered in the message handling. As the drive is not essential for the operation of the machine, it should be deselected from the message handling and from the customer application. After setting the property, no more errors from these objects are entered in the message handling.

Example 2

Partial commissioning of a machine

The hardware of a machine is not complete at the start of commissioning, e.g. a motor and an I/O node are missing. Entries for these objects are to be suppressed in the message handling.

Suppressible messages on objects

The following objects can be suppressed:

- Technology objects
- Drive objects
- I/O modules

Each object is assigned a setting in which you define whether the message handling should monitor the object or not. These settings are saved retentively by the message handling and are therefore still present after Power OFF/ON.

There are two ways to specify the objects that are to be monitored.

- First of all, a basic specification can be set in the source code of the message handling. This is performed in the **fLMsgHdlInit** program unit in the **FCLMsgHdlInitProjectInfo** function.
- In addition, this basic setting can be changed or adapted by the user during runtime. The setting changed during runtime overwrites the basic setting and is then valid at each restart. The basic settings are taken over by the message handling either when the configuration script is run through again, or when the user increments the global constant LMSGHDL_SCRIPT_COUNTER in the source code of the **fLMsgHdlInit** program unit, recompiles and downloads the project to the SIMOTION device.

Basic settings in the source code

The basic settings in the source code are made as follows:

- TO axis with DO
  If the TO is to be monitored, this is transferred in **fLMsgHdlInit** in the **FCLMsgHdlInitProjectInfo** function via *gasLMsgHdlaxes[numberOfAxis].boToUsedInProject := TRUE*. The associated DO is transferred via *gasLMsgHdlaxes[numberOfAxis].boRelatedDoUsed := TRUE*.
- All other TOs
  Within the transfer structure of all TO types there is a setting with which the monitoring can be switched on or off for each TO, e.g. *gasLMsgHdlExternalEncoders[numberOfTO].boToUsedInProject := TRUE.*
- DO with cyclic data exchange
  Within the transfer structure of all DOs with cyclic data exchange there is a setting with which the monitoring can be switched on or off for each DO, e.g. *gasLMsgHdlDOsCyclic[numberOfCyclicDO].boDoUsedInProject := TRUE.*
- DO without cyclic data exchange
  Within the transfer structure of all DOs without cyclic data exchange there is a setting with which the monitoring can be switched on or off for each DO, e.g. *gasLMsgHdlDOsACyclic[numberOfCyclicDO].boDoUsedInProject := TRUE.*
- I/O modules
  Within the transfer structure of all I/O modules there is a setting with which the monitoring can be switched on or off for each I/O module, e.g. *gasLMsgHdlPeripheralDevices[numberOfDevice].boUsedInProject := TRUE.* If *boUsedInProject := FALSE* is set for an I/O module, it is essential that the module also does not actually exist. If it is present however, an error is output by the message handling. When deselecting a SINAMICS I/O module, no time synchronization is performed on this module.

Note

If an I/O module of the SINAMICS type is not available, all objects belonging to the device must be deselected. This means that the corresponding property must be set to FALSE for all TOs, all DOs and the I/O modules themselves. If this is not performed, error messages are output by the message handling.

Settings during runtime

The settings during runtime are performed as follows:

The settings described here are stored within the **pLMsgHdl** program unit in the retain data.

- TO axis with DO
  The *gsLMsgHdlMoMaTOAxis* variable is of the *sLMsgHdlMoMaTOAxisType* type and is in the **pLMsgHdl** program unit. This means that the settings for modular machines can also be made via the symbol browser.
  The information, as to which entries belong to which technology object, is transferred by the message handling.

The data type is defined as follows:

Table 3- 8 Structure of sLMsgHdlMoMaTOAxisType

Parameter	Data type	Initial value	Description
toReference	ANYOBJECT	TO#NIL	The axis reference is set from the execution software of the message handling.
sgToName	STRING		Name of the axis
boRelatedDOUsed	BOOL	TRUE	With FALSE, the DO belonging to the TO is not monitored by the message handling.
boTOUsedInProject	BOOL	TRUE	With FALSE, the TO is not monitored by the message handling.

- All other TOs
  The *gsLMsgHdlMoMaxxx* variables (type of the TO) are of the *sLMsgHdlMoMaTOType* type and are in the **pLMsgHdl** program unit. This means that the settings for modular machines can also be made via the symbol browser.
  The information, as to which entries belong to which technology object, is transferred by the message handling.
  The following variables are available depending on the active technology package:
  - *gasLMsgHdlMoMaExternalEncoders* for external encoders
  - *gasLMsgHdlMoMaMeasuringInputs* for measuring inputs
  - *gasLMSGHDLMoMaOutputCams* for output cams
  - *gasLMsgHdlMoMaCamTracks* for cam tracks
  - *gasLMsgHdlMoMaCams* for cams
  - *gasLMsgHdlMoMaFollowingObjects* for following objects
  - *gasLMsgHdlMoMaPathObjects* for path objects
  - *gasLMsgHdlMoMaFixedGears* for fixed gears
  - *gasLMsgHdlMoMaAdditionObjects* for addition objects
  - *gasLMsgHdlMoMaFormulaObjects* for formula objects
  - *gasLMsgHdlMoMaSensors* for sensors
  - *gasLMsgHdlMoMaControllerObjects* for controller objects
  - *gasLMsgHdlMoMaTemperatureControllers* for temperature controllers

The data type is defined as follows:

Table 3- 9 Structure of sLMsgHdlMoMaTOType

Parameter	Data type	Initial value	Description
toReference	ANYOBJECT	TO#NIL	The axis reference is set from the execution software of the message handling.
sgToName	STRING		Name of the axis
boTOUsedInProject	BOOL	TRUE	With FALSE, the TO is not monitored by the message handling.

- DO with cyclic data exchange
  The *gasLMsgHdlMoMaDosCyclic* variable is of the *sLMsgHdlMoMaDOsCyclicType* type and is in the **pLMsgHdl** program unit. This means that the settings for modular machines can also be made via the symbol browser.
  The information, as to which entries belong to which drive object, is transferred by the message handling.

The data type is defined as follows:

Table 3- 10 Structure of sLMsgHdlMoMaDOsCyclicType

Parameter	Data type	Initial value	Description
sgDoName	STRING		Name of the DO
i32LogAddress	DINT	0	Logical address of the DO
eIoId	enumIoIdType	INPUT	Data direction of the logical address
boDOUsedInProject	BOOL	TRUE	With FALSE, the DO is not monitored by the message handling.

- DO without cyclic data exchange
  The *gasLMsgHdlMoMaDosAcyclic* variable is of the *sLMsgHdlMoMaDOsACyclicType* type and is in the **pLMsgHdl** program unit. This means that the settings for modular machines can also be made via the symbol browser. The information, as to which entries belong to which drive object, is transferred by the message handling.

The data type is defined as follows:

Table 3- 11 Structure of sLMsgHdlMoMaDOsACyclicType

Parameter	Data type	Initial value	Description
sgDoName	STRING		Name of the DO
i32LogAddress	DINT	0	Logical address of the DO
eIoId	enumIoIdType	INPUT	Data direction of the logical address
u8DoNumber	USINT	0	DO number
boDOUsedInProject	BOOL	TRUE	With FALSE, the DO is not monitored by the message handling.

- I/O modules
  The *gasLMsgHdlMoMaPeripheralDevices* variable is of the

sLMsgHdlMoMaPeripheralDevicesType type and is in the **pLMsgHdl** program unit. This means that the settings for modular machines can also be made via the symbol browser. The information, as to which entries belong to which I/O module, is transferred by the message handling.

The data type is defined as follows:

Table 3- 12 Structure of sLMsgHdlMoMaPeripheralDevicesType

Parameter	Data type	Initial value	Description
sgDeviceName	STRING		Name of the I/O module
u32MasterSystemId	UDINT	0	*masterSystemId* of the bus system on which the I/O module has been configured.
u32SlaveAddress	UDINT	0	Slave address of the I/O module
boUsedInProject	BOOL	TRUE	With FALSE, the I/O module is not monitored by the message handling. It must not be present and is not time-synchronized (SINAMICS module).

The transfer or validity of this information is performed by setting *gboLMsgHdlActivateNewMoMaData* in the **pLMsgHdl** program unit. After the transfer, the setting *gboLMsgHdlActivateNewMoMaData* is removed automatically. This setting is then valid after each Power OFF/ON restart.

3.4.2.11 DO safety messages

General

With the message handling, it is possible to also display active safety messages for the DO errors and warnings in the message handling. So that the safety messages can be read out at a DO, the *eCheckSIMessages* additional information must be supplied with the value BY_DO_ADDRESS in the **fLMsgHdlInit** unit for each DO with active safety configuration. As presently there is no bit for active safety messages in the cyclic data exchange, it must be read out directly via the acyclic data exchange in a DO with active safety whether safety messages are active or not. As to whether safety messages are active or not has to be read out acyclically each time, *eCheckSIMessages* should only be set for those DOs on which safety messages can occur. DISABLED should be set for all other DOs (default).

If a safety message occurs on a DO, it is entered in the message buffer. Safety messages have approximately the same behavior as warnings on the DO. This means that safety messages cannot be acknowledged via the message handling. When a safety message is no longer active on the DO, this is detected by the message handling and automatically acknowledged with the corresponding time stamp in the message handling.

The information as to whether safety messages are to be monitored on a DO can be activated with the three different DO types at the following locations:

DO with TO

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

DO with cyclic data exchange

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

DO without cyclic data exchange

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

Table 3- 13 Enum for eLDPV1CheckSIMessagesType

Enum identifier	Description
DISABLED (0)	Check of safety messages switched off on the DO
BY_DO_ADDRESS (2)	Check whether safety messages are active via acyclic read job

Note

When using safety, the messages are automatically acquired by the message handling.

You do not have to make the settings described here.

BY_DO_ADDRESS is preset.

3.4.2.12 Saving of the ShutdownTask buffer

In the message handling, the buffer for user-defined messages within the ShutdownTask is non-volatile. In this way, user-defined messages of the ShutdownTask are displayed in the message handling when the machine is restarted.

3.4.2.13 Saving the current message log in the SIMOTION device

Saving of the message log

There are two different ways to save the message log to the storage medium of the SIMOTION device. Both options can be used together.

Note

The message log is saved as soon as the buffer is full with the *AutoSave* function. All messages that do not have a *gone* time stamp at this time are saved without this time stamp.

Option 1

It is possible to save the current buffer in raw data format and STRING format to the storage medium of the SIMOTION device. To save the buffer, the global variable *gboLMsgHdlStartWriteCompleteMessageLogToStorageMedium* must be set to TRUE in the **pLMsgHdl** program unit. The name of the file in which the information is written, is set via the *gu32LMsgHdlDataSetNoForExportMessageLog* variable (default is 0). The rising edge saves the current message buffer to the storage medium of the SIMOTION device. All the information that is required to be able to interpret the message log in raw data format is also contained in the file. After saving, the global variable is reset to FALSE. After this action, the data of the buffer is on the storage medium of the SIMOTION device in the following directory:

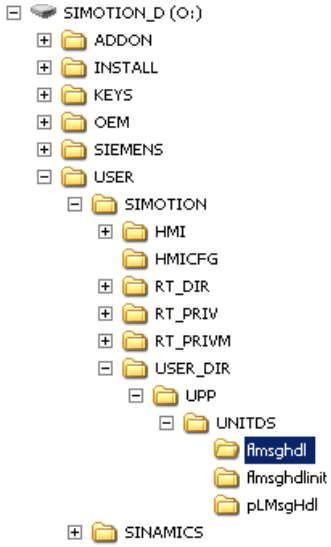

Figure 3-2 fLMsgHdl directory

In the **fLMsgHdl** directory, the file with the buffer images is called *dsxxxxxx.dat*, whereby *xxxxxx* corresponds to the number in *gu32LMsgHdlDatasetNoForExportMessageLog*. A diagnosis is possible based on this file.

Option 2

The error history can be saved permanently to the storage medium of the SIMOTION device. Saving is performed automatically when the log buffer in the SIMOTION device is full.

The number of data sets for backing up the data is specified by the user via the constant in front of LMSGHDL_MAX_NUMBER_OF_DATASETS_ON_STORAGE_MEDIUM in the **cPublic** unit. The data sets are created according to the principle of a ring buffer. The functionality can be switched on or off via LMSGHDL_AUTO_SAVE_MESSAGE_BUFFER_TO_STORAGE_MEDIUM in the **cPublic** unit.

Properties of the functionality:

The automatic saving is activated via message handling constants in the **cPublic** unit. The user also specifies the number of data sets or the size of the ring buffer in the **cPublic** unit, e.g. 3.

If an overflow occurs in the ring buffer of the message handling, the buffer data is saved consistently to the storage medium of the SIMOTION device. The information stored there is assigned the data set numbers 1000/1001/1002 etc. When the maximum number of data sets has been reached, the first data set is overwritten in the ring buffer operation.

3.4.2.14 Loading the language from the storage medium of the SIMOTION device

Proceed as follows to load different languages:

The value set in the constant LMSGHDL_LANGUAGE_FOR_MESSAGE_STRING in the **cPublic** unit of the **LMsgHdl** library specifies with which language the message handling starts after startup. The value of the respective language corresponds to the language ID from the STEP 7 notation. German, English, French and Italian are currently integrated. Other languages must be created and loaded by the user. Each time the SIMOTION device is started, the message handling loads the appropriate language files for the system messages and user-defined messages from the storage medium of the SIMOTION device to the message handling. If these language files are not available, the language (German or English) set via the configuration script is used by the message handling.

During message handling operation, the language selection can be changed as desired between the languages stored on the storage medium of the SIMOTION device. This is performed via the variables *gu8LMsgHdlActiveLanguage* and *gboLMsgHdlStartChangeLanguage* from the **pLMsgHdl** unit. The language to be used is transferred as language ID to the *gu8LMsgHdlActiveLanguage* variable and exchanged with a rising edge in the *gboLMsgHdlStartChangeLanguage* variable. If the appropriate language files are available on the storage medium of the SIMOTION device, all message buffers in the STRING format are recreated after loading the language messages. The message buffers are then output directly in the changed language. If an error occurs during loading, e.g. the selected language is not available on the storage medium of the SIMOTION device, a message is entered in the message handling.

The files with the system messages in the various languages must be stored on the storage medium of the SIMOTION device in the following directory:
USER/SIMOTION/USER_DIR/UPP/UNITDS/pLMsgHdl

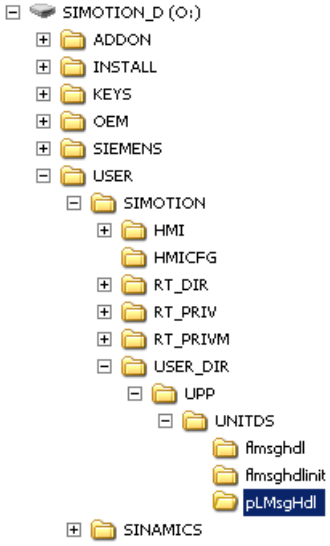

Figure 3-3 Directory for language files

Figure 3-4 Language files

The file names of the various languages are formed from the abbreviation **ds** together with the language ID from the STEP 7 notation for the respective language. Consequently, the *ds000007.dat* file shown above in the **pLMsgHdl** directory contains all the system message texts in German.

The files with the user-defined messages in the various languages must be stored on the storage medium of the SIMOTION device in the following directory:
USER/SIMOTION/USER_DIR/UPP/UNITDS/fLMsgHdlInit

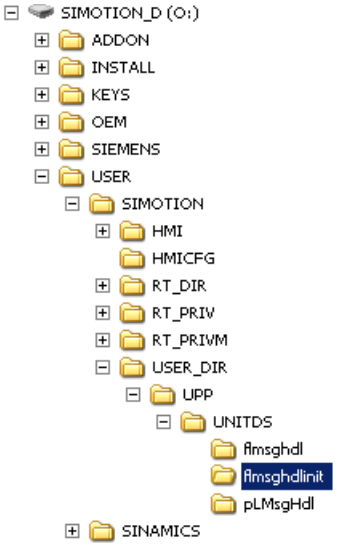

Figure 3-5 Directory for languages of the user-defined messages

Figure 3-6 Files for languages of the user-defined messages

The file names of the various languages are also formed from the abbreviation **ds** and the language ID from the STEP 7 notation for the respective language. Consequently, the *ds000007.dat* file shown above in the **fLMsgHdlInit** directory contains all the user-defined message texts in German.

Table 3- 14 Used language codes according to the STEP 7 notation (language ID)

Language	**Value [USINT]**
No message log in STRING format	0
German	7
English	9

Language	Value [USINT]
Spanish	10
French	12
Italian	16

3.4.2.15 Single acknowledgement

Single acknowledgement of messages

General

In addition to the global acknowledgement of all active messages, all active messages of one source or even single messages can be acknowledged, depending on the message source.

In the **pLMsgHdl** program unit, the *gi32LMsgHdlNumberOfMessageInLog* variable informs the message handling which message is to be acknowledged. The acknowledgement itself is started via *gboLMsgHdlGlobalAcknowledge* as for the global acknowledgement.

If all active messages are to be acknowledged globally, a -1 must be entered in *gi32LMsgHdlNumberOfMessageInLog* (default).

With single acknowledgement, the number of the message to be acknowledged results from the number of the message in the buffer for active messages. This means that if the first active message from the message buffer is to be acknowledged, the number 1 must be entered before starting the acknowledgement in *gi32LMsgHdlNumberOfMessageInLog*. After the acknowledgement, the contents of this variable is again set to -1 and *gboLMsgHdlGlobalAcknowledge* reset to FALSE.

Single acknowledgement for the message sources

- TO messages
  Single acknowledgement of a TO message broken down to an entire TO, at which errors are pending, e.g. *_resetAxisError(ALL_ERRORS)*. All the pending messages are acknowledged for the selected TO.
- DO messages
  Only DO errors can be acknowledged. DO warnings or safety messages cannot be acknowledged, but disappear automatically as soon as the reason for the message no longer applies. If an error is to be acknowledged on a DO, a single acknowledgement of the relevant DO is triggered. All other errors belonging to this DO are therefore also acknowledged. If the errors are still pending after acknowledgement, they are entered again in the message handling.
- I/O messages
  Only the selected I/O message is acknowledged. The I/O messages with the identifiers 202, 204, 210 and 215 cannot be acknowledged. These messages are automatically acknowledged with the appropriate counter-message.
- TimeFault messages
  Only the selected TimeFault message is acknowledged.

- Messages by message handling
  Only the message selected by the message handling is acknowledged.
- User-defined messages
  Each user-defined message is individually acknowledged. The single acknowledgement also influences the bit message procedure and ALARM_S.

Selection of the message to be acknowledged via SIMOTION IT

On the SIMOTION IT page, the user can select single, active messages that are to be acknowledged. The use of buttons for the single acknowledgement ensures that only a single message or message source is actually acknowledged. After the selection has been made, this selection is transferred to the message handling when the **Acknowledge** button belonging to the message is clicked. The message handling then performs the single acknowledgement. The information required by the message handling is available in *ai32NumberOfMessageInLog[LineInHMI - 1]* in the *gsLMsgHdlActiveMsgToHMI* variable of the **pLMsgHdl** unit. This information is transferred for single acknowledgement to *gi32LMsgHdlNumberOfMessageInLog* of the **pLMsgHdl** unit. At the same time *gboLMsgHdlGlobalAcknowledge* is assigned a rising edge by SIMOTION IT.

User-defined message based on the message number

To be able to acknowledge a "single" user-defined message, a new global variable for transfer of the message number has been inserted in the **pLMsgHdl** program unit.

The variable for the transfer of the number is *gi32LMsgHdlNumberOfUserMessage*.

To acknowledge a specific user-defined message, the number of the message must be entered in this variable. The acknowledgement is then made with the rising edge at *gboLMsgHdlGlobalAcknowledge*.

Since the user-defined messages also support the bit message procedure and AlarmS procedure, the messages of these procedures are also acknowledged during the single acknowledgement.

By specifying the message number, it is thus possible to "specifically" acknowledge user-defined messages, user-defined messages of FBs/FCs, and messages by the message handling itself.

Since the user-defined message with the message number 0 may be active several times, make sure that only the first entry found with the identifier 0 is acknowledged with the single acknowledgement. However, if all alarms with the identifier 0 are to be acknowledged simultaneously, the acknowledgement procedure described in 1.2 should be used.

Note

It is recommended that other single acknowledgement mechanisms (Page 45) are used since shifts in the buffer can occur between selection of the message number and issuing of the acknowledgement in the procedure described here, which may result in the wrong message being acknowledged.

Group acknowledgement of certain sources

In order to be able to acknowledge all messages of a certain source, e.g. of a certain DO, the *gsLMsgHdlAcknMessageInfo* variable of the *sLMsgHdlAcknMessageInfoType* type has been inserted in the **pLMsgHdl** program unit. In this way, it is possible to acknowledge all messages that satisfy the possible criteria at the same time.

The options of this type of acknowledgement are described below in more detail.

Table 3- 15 Elements of the *sLMsgHdlAcknMessageInfoType* structure

Element	Data type	Initialization value	Meaning
u8MessageSource	USINT	0	Source of messages to be acknowledged in the message handling
u16Parameter1	UINT	0	Value to identify message 1 to be acknowledged. (See Appendix B. 1 for the interpretation of the raw data)
i16Parameter2	INT	0	Value to identify message 2 to be acknowledged. (See Appendix B. 1 for the interpretation of the raw data)
b32Parameter3	REAL	16#FFFF_FFFF	Value to identify message 3 to be acknowledged. (See Appendix B. 1 for the interpretation of the raw data)
b32Parameter4	REAL	255	Value to identify message 4 to be acknowledged. (See Appendix B. 1 for the interpretation of the raw data)
b32Parameter10	REAL	0	Value to identify message 10 to be acknowledged. (See Appendix B. 1 for the interpretation of the raw data)

The elements of the structure for transfer of the acknowledgement information correspond to the data entered by the different message types in the raw data buffer of the message handling. For more detailed information, refer to Appendix B. 1 of the documentation for the message handling.

Note

The acknowledgement procedure described here works in principle according to the method that the information requested in the acknowledgement structure is taken 1:1 from the raw data buffer.

Messages from technology objects

All the active errors of an individual technology object can be acknowledged here. If the technology object is an axis with a real drive, the subordinate drive object is also acknowledged.

In order to acknowledge a specific technology object, the following data must be transferred to *gsLMsgHdlAcknMessageInfo* .

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 1;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= identifier of the corresponding TO type
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= TO number
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= not relevant

gsLMsgHdlAcknMessageInfo.b32Parameter4 := not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10 := not relevant

The TO number corresponds to the respective subindex under which the technology object has been entered by the script in **fLMsgHdlInit** .

The respective identifiers for the TO types are listed in the Appendix (Page 120).

Errors on drive objects

All the active errors of an individual drive object (DO) can be acknowledged here.

In order to acknowledge a specific drive object, the following data must be transferred to *gsLMsgHdlAcknMessageInfo* .

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= TO number (for DO with TO)
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= IoId (INPUT/OUTPUT)
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= logical address for the DO
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= DO number
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= not relevant

The following must be taken into account:

If the drive object is a real drive that is assigned to an axis, the subindex of the axis must first be entered in parameter 1. Parameters 2-4 **must** be set to their default values. Parameter 10 is not relevant.

Example:

Acknowledgement of the drive object for the axis

gasLMsgHdlAxes[1].toReference := film take-off;

gsLMsgHdlAcknMessageInfo.au8MessageSource	:= 2;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= 1
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= 0
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= 16#FFFF_FFFF
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= 255
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= not relevant

If the drive object is a DO with cyclic data exchange, default value 0 must be set in parameter 1. Parameters 2-3 **must** be assigned the values for the IoId and the associated logical address of the telegram. Parameter 4 has the DO number 255 and parameter 10 is not relevant.

Example:

Acknowledgement of the CU with telegram 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     := not relevant
```

If the drive object is a DO without cyclic data exchange, default value 0 must be set in parameter 1. Parameters 2-4 **must** be assigned the values for the IoId, an associated logical address and the DO number. Parameter 10 is not relevant.

Example:

Acknowledgement of the DO of the TB30 without its own telegram

```
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     := not relevant
```

Messages from I/O modules

With the transfer of *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 4
all active messages through I/O modules that can be acknowledged are canceled.

The messages from I/O modules with interrupt IDs 202, 204, 210 and 215 cannot be acknowledged. They "disappear" automatically as soon as the associated I/O message that the error is no longer pending has been detected.

Messages through timeouts

With the transfer of *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 5
all active messages generated through timeouts in timer tasks or the background task are acknowledged.

Messages of the execution fault task after restart of the controller

With the transfer of *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 6
all active messages generated through the call of the execution fault task are acknowledged.

User-defined messages

User-defined messages can be acknowledged individually as follows:

gsLMsgHdlAcknMessageIn-fo.au8MessageSource	:= 9;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= not relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= message number to be acknowledged
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= not relevant

If message number 0 is transferred in parameter 3, all the active messages with message number 0 are acknowledged.

With the acknowledgement of user-defined messages, the associated bit messages and AlarmS messages are also acknowledged, if used.

Messages from FBs/FCs

User-defined messages from FBs/FCs can be acknowledged individually as follows.

gsLMsgHdlAcknMessageIn-fo.au8MessageSource	:= 8;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= not relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= message number to be acknowledged
gsLMsgHdlAcknMessageInfo.b32Parameter4	:= not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10	:= not relevant

With the acknowledgement of user-defined messages, the associated bit messages and AlarmS messages are also acknowledged, if used.

In addition, it is also still possible to acknowledge all active messages from FBs/FCs with the same function block ID at the same time. In this case, all messages, regardless of the message number, are acknowledged in transfer parameter 10 depending on the function block ID.

gsLMsgHdlAcknMessageIn-fo.au8MessageSource	:= 8;
gsLMsgHdlAcknMessageInfo.u16Parameter1	:= not relevant
gsLMsgHdlAcknMessageInfo.i16Parameter2	:= not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter3	:= not relevant

gsLMsgHdlAcknMessageInfo.b32Parameter4 := not relevant
gsLMsgHdlAcknMessageInfo.b32Parameter10 := 5

In this way, all messages through FBs/FCs with identifier 5 in parameter 10 (functionBlockId) would be acknowledged simultaneously.

Messages through the message handling

With the transfer of *gsLMsgHdlAcknMessageInfo.au8MessageSource* := 10
all active messages generated by the message handling itself are acknowledged.

3.4.2.16 Common buffer for incoming/outgoing messages

Functionality

In order to be able to use the new buffer in raw data format, the **dLMsgHdl** program unit must first be imported into the SIMOTION project. At present, this import is not supported by the script and must be performed separately via the XML import of a program unit.

After the import, the LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED define commented out, still has to be commented in in **dLMsgHdl**. This define must also be commented in in the **cPublic** library unit of the **LMsgHdl** library.

Only after the define has been commented in in both units and the project recompiled, is the new message buffer active and can be used.

In the **fLMsgHdlInit** program unit, the *USES dLMsgHdl* code line also has to be commented in and the *USELIB LMsgHdl* code line commented out.

The *LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURRED* constant determines how many entries can be included in the buffer. Since this buffer is a ring buffer, the oldest message is overwritten in the event of an overflow.

The constant is in the **cPublic** library unit and must be set by the user to the required value. The default value of the constant is 200.

If this buffer is to be accessed by means of the HMI, also ensure that the size of the buffer does not exceed 64k.

Structure of the buffer

Table 3- 16 Elements of the *sLMsgHdlMessageLogBaseDataGoneAndOccurredType* structure

Element	Data type	Meaning
i16ActualIndex	INT	Current index in the global message log for raw data
au8MessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array for information about the source of the message
au8MessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array for information about the level of the message (fault, alarm, error, information)

Element	Data type	Meaning
au8AcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array for information about the type of acknowledgement for the message
au8ErrorClass	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF USINT	Array for information about the error class of a message
aboOccuredMessage	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF BOOL	Array for information about whether the message is an incoming or outgoing message TRUE indicates that it is an incoming message
au16Parameter1	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF UINT	Array for information about variable 1 of the respective message type
ai16Parameter2	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF INT	Array for information about variable 2 of the respective message type
ab32Parameter3	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 3 of the respective message type
ab32Parameter4	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 4 of the respective message type
ab32Parameter5	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 5 of the respective message type
ab32Parameter6	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 6 of the respective message type
ab32Parameter7	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 7 of the respective message type
ab32Parameter8	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 8 of the respective message type
ab32Parameter9	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 9 of the respective message type
ab32Parameter10	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 10 of the respective message type
ab32Parameter11	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DWORD	Array for information about variable 11 of the respective message type
adtMessageGoneOccured	ARRAY[0..LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURED - 1] OF DT	Time stamp when the relevant message has occurred or has gone
dLMsgHdl	LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED	Alternative message variant (not activated by default)

The *LMSGHDL_LENGTH_OF_MESSAGE_LOG_GONE_OCCURRED* constant determines how many entries can be included in the buffer. Since this buffer is a ring buffer, the oldest message is overwritten in the event of an overflow.

The constant is in the **cPublic** library unit and must be set by the user to the required value. The default value of the constant is 200.

If this buffer is to be accessed by means of the HMI, also ensure that the size of the buffer does not exceed 64k.

Integration

4

4.1 Required technology objects

The CAM technology package is a minimum requirement for the message handling. Depending on the technology objects used, the message handling can also be operated with the TP Path, TP Cam_ext and TControl.

4.2 Integration in the SIMOTION project

4.2.1 Integration of the application into a SIMOTION project

The **LDPV1** and **LMsgHdl** libraries can be integrated into the existing project via the **ProjectGenerator** which is part of the scope of delivery of SIMOTION SCOUT (on the Utilities & Applications storage medium).

Proceed as follows to execute the ProjectGenerator:

1. Close SMOTION SCOUT.
2. Double-click the **ProjectGenerator.exe** file.
3. Confirm the disclaimer of liability and select the **Create a new project** option.

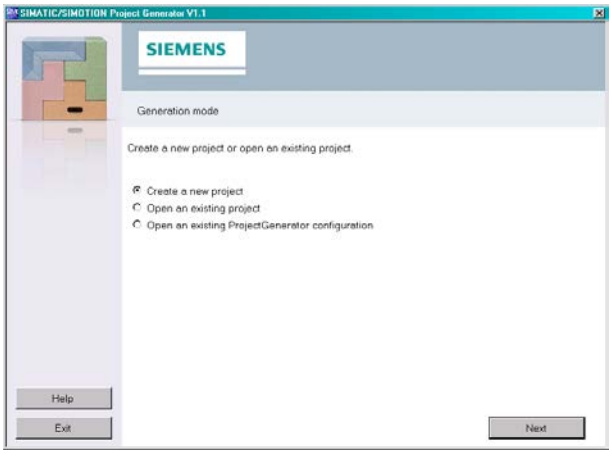

Figure 4-1 Creating or selecting a project

4. If you have selected **Create a new project**, enter a project name and the storage location of the project (the path can also be selected via **Browse path**) and click **Next**.

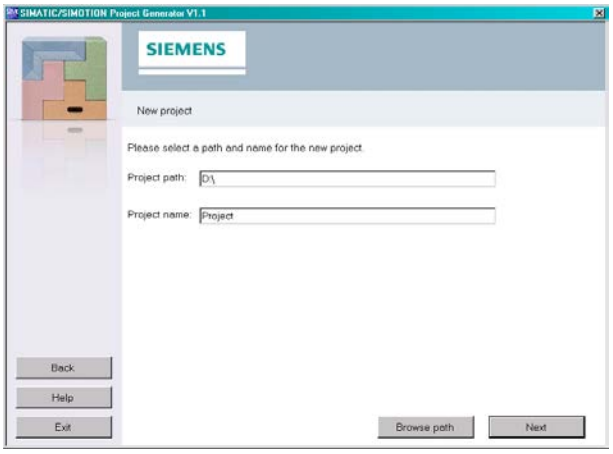

Figure 4-2 Project name and storage path for the project

The *SIMATIC/SIMOTION project data - device selection* window is opened.

5. In the *SIMATIC/SIMOTION project data - device selection* window, at **Select device category**, select which device or devices should be integrated into the project:

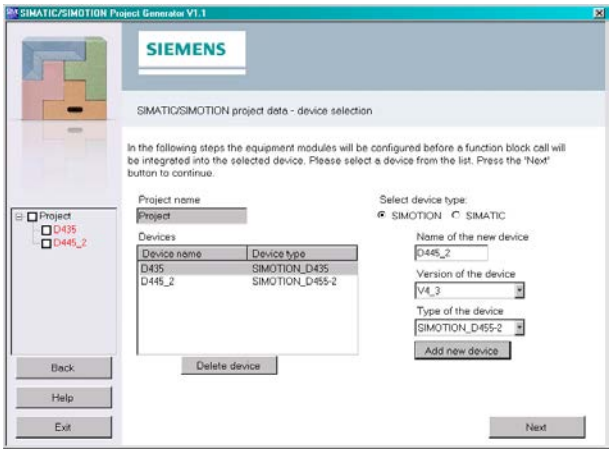

Figure 4-3 Creating or selecting a device

Depending on the software installation, you can select either only SIMATIC devices or mixed SIMOTION and SIMATIC devices, one after the other.
If a device has not been created yet, you can create a new device on the right-hand side of the window. To do this, enter the *Device name*, the *Version* and the *Type name* of the device and click **Add new device**. The new device is taken into the **Devices** list.
If you want to create a further device, repeat the procedure.

6. In the **Devices** list, select the device you want to configure and click **Next**.
The *SIMATIC/SIMOTION project data - equipment module selection* window opens.

7. In the *SIMATIC/SIMOTION project data - equipment module selection* window, select the standard module or modules that you want to integrate into the selected device, in this case **Message Handling**, and click **Next**.

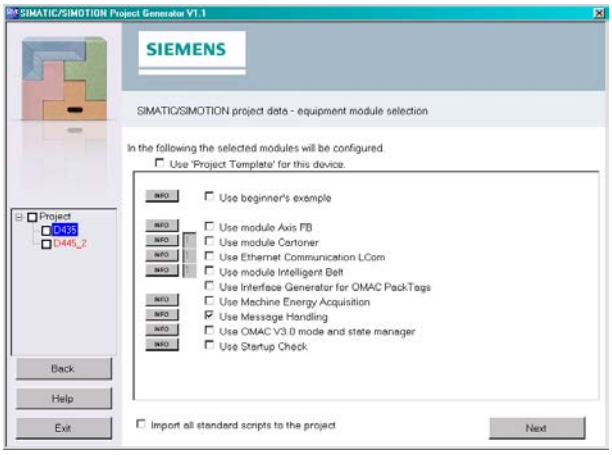

Figure 4-4 Module selection

Click the **INFO** button to open the documentation for the respective standard module. For some standard modules, you can enter the number of modules on the left.

8. In the *Message Handling - Configuration* window, configure the call of the function block with the required data blocks and data exchange parameters and click **Next**.

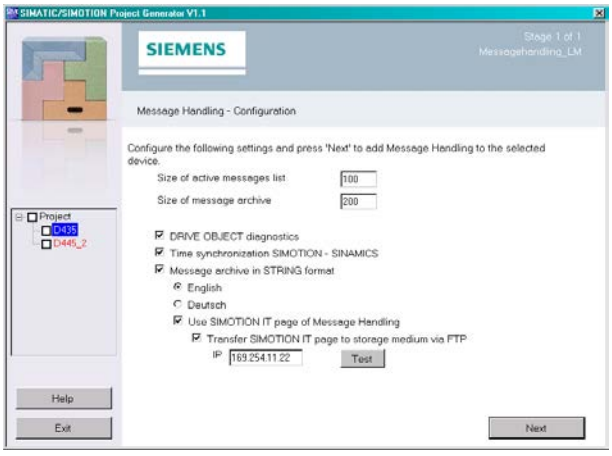

Figure 4-5 Message handling configuration

9. Configure all the other devices following the above example by clicking the **Configure another device** button.

   Fully configured devices are shown in green with a checkmark on the outer left beneath the project name, while devices that have not yet been configured are red, and devices being worked on are orange.

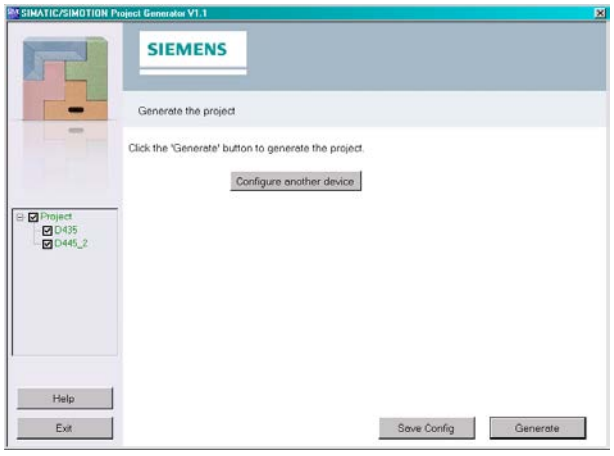

Figure 4-6 Generating the project

10. If you do not want to configure another device (**Configure another device**) and want to complete the project, you have the following options:

   – If you would like to save the configuration of the devices, but generate the project at a later time, click **Save Config** and enter the storage path in Explorer.

   – If you do not wish to configure any other devices and would like to generate the project, click **Generate**.
   The project is generated. The duration depends on the type of configuration and is shown using a progress bar.
   When the project has been completely generated, the message **Generation finished** appears in the window.

11. Click the **Exit** button to close the ProjectGenerator.

The ProjectGenerator has been completed and returns the following result:

Table 4- 1 Result after execution of the configuration script

Result	Remark
Libraries LDPV1 and LMsgHdl have been imported into the SIMOTION project	A version check was performed automatically. If the versions do not match, a message is generated and the user can decide whether the libraries are to be replaced or not.
ST units have been created in the SIMOTION project: • fLMsgHdlInit • fLMsgHdl • pLMsgHdl • pGlobalBufferManager	Configuration of the TOs, drives, I/O modules, etc.

Result	Remark
ST units were assigned to various execution system tasks of the SIMOTION project	**StartupTask**: pLMsgHdlStartupMessagehandling **BackgroundTask**: pLMsgHdlMain, pGlobalBufferManager **TimeFaultTasks**: pLMsgHdlTimeFaultMessage **TimeFaultBackgroundTask**: pLMsgHdlTimeFaultBackgroundMessage **TechnologicalFaultTask**: pLMsgHdlTechnologicalMessage **PeripheralFaultTask**: pLMsgHdlPeripheralMessage **ExecutionFaultTask**: pLMsgHdlExecutionFaultMessage
A compilable project has been saved	
Data for the use of the control panel via SIMOTION IT has been saved	Depending on the selection, either on the PC or on the storage medium of the SIMOTION device.

The ProjectGenerator automates various actions in the SIMOTION project and simplifies the commissioning of the message handling. The adaptations of the properties of the task system are changed by the ProjectGenerator. The libraries can also be integrated via the **Import library** SIMOTION SCOUT function. In this case, the user must perform all the required actions individually so that the message handling can be used.

Subsequent transfer of the SIMOTION IT page to the storage medium of the SIMOTION device

If the **Transfer SIMOTION IT web page to storage medium via FTP** function has not been selected when running through the ProjectGenerator, the data is saved in the following path in the project directory: **C:\Documents and Settings\<Login name>\Local Settings\Temp\LMsgHdl_Files_for_HMI\PGEN_Data_Files\CardFiles**. To be able to use this data, it must be saved to the storage medium of the SIMOTION device.

As of Version 1.3.0 of the ProjectGenerator, in addition to the previous possibility of transferring SIMOTION IT pages via FTP to a controller during the generation, it is now possible to transfer only the SIMOTION IT pages of an existing project to a controller, without having to regenerate the project A detailed description of the subsequent transfer of the SIMOTION IT pages can be found in Section *Transfer of SIMOTION IT pages* in the *SIMATIC/SIMOTION ProjectGenerator* Application Manual.

4.2.2 Suppressing messages

Variance in the message handling by setting defines

By setting preprocessor definitions (defines) in the properties of the **pLMsgHdl** and **fLMsgHdl** program units, it is possible to suppress certain functionalities in the message handling outside of the variance performed by the script.

The defines can be entered via the context menu (right-click) on the unit **pLMsgHdl -> Properties** in the **Further settings** tab. As the unit is write-protected, a window is displayed for the password query. Click the **Cancel** button. The ST Source File Properties window opens. Detailed information about this function can also be found in the SIMOTION online help at Preprocessor -> Changing properties of an ST source file.

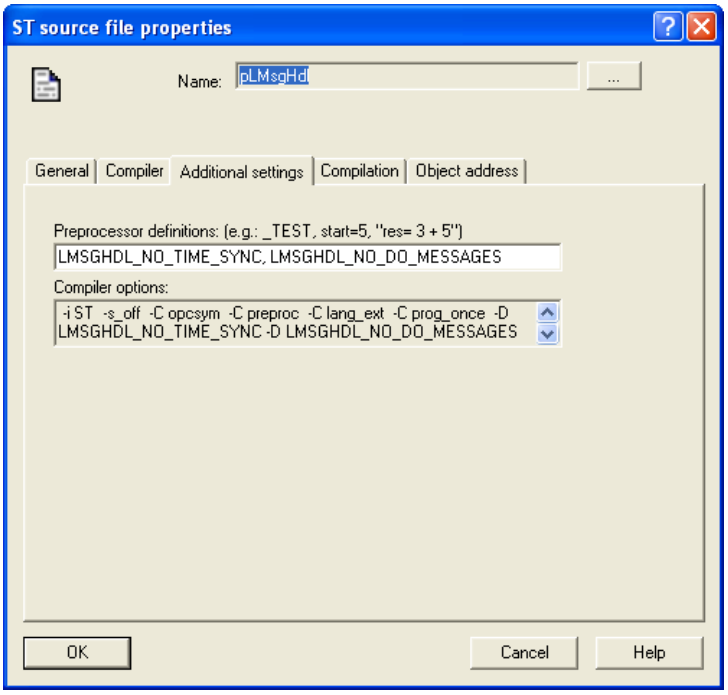

Figure 4-7 Suppressing messages (example)

After preprocessor definitions have been set, the program unit must be recompiled.

The following selection options are available in the **pLMsgHdl** program unit:

- LMSGHDL_NO_TECH_FAULT_MESSAGES
  No messages through technology objects are evaluated in the TechnologicalFaultTask by the message handling.
- LMSGHDL_NO_PERIPHERAL_MESSAGES
  No messages through I/O modules are evaluated in the PeripheralFaultTask by the message handling.
- LMSGHDL_NO_TIME_FAULT_MESSAGES
  No messages through timeouts are evaluated in the TimeFaultTask and the TimeFaultBackgroundTask by the message handling.
- LMSGHDL_NO_EXECUTION_FAULT_MESSAGES
  No messages through program faults are evaluated in the ExecutionFaultTask by the message handling.
- LMSGHDL_NO_DO_MESSAGES
  No messages on drive objects on all SINAMICS modules are evaluated by the message handling.
- LMSGHDL_NO_TIME_SYNC
  The time synchronization of the SINAMICS modules is not carried out.
- LMSGHDL_NO_HMI_FBS
  The program to transfer the message buffers to the HMI / SIMOTION IT is not called.
- LMSGHDL_NO_STRING_MESSAGES
  The program to transfer the message buffers to the HMI / SIMOTION IT transfers the information of the raw data buffer.

The following selection option is available in the **fLMsgHdl** program unit:

- LMSGHDL_NO_RETAIN_BUFFER
  Change of the message log data area of the SIMOTION device from RETAIN to NON-RETAIN.

The following selection option is available in the **dLMsgHdl** program unit:

- LMSGHDL_BUFFER_FOR_MESSAGES_GONE_AND_OCCURRED
  Alternative message variant

4.2.3 Creating user-defined messages

Integrating user-defined messages in the STRING format into the message handling

Users can issue user-defined messages from their own application, which are entered in the buffers of the message handling. These messages are treated the same as messages from the system. In order to be able to correctly assign the messages, the message is allocated a defined identification number (ID) by the user. The messages can also be allocated a machine error class, see also Section Defining machine error classes (Page 62).

The messages defined by the user are entered in the **fLMsgHdlInit** unit. The rules for generating the messages are explained in the following:

- All user-defined messages are transferred to the *gasLMsgHdlUserDefinedMessages* global variable in the **FCLMsgHdlUserDefinedInfoForMessageHandling** function of the **fLMsgHdlInit** unit.
- The number of user-defined messages is in the range from 1 to 99.999. The messages start with the subindex 0.
- Because of a faster runtime when generating the message texts in STRING format, the message texts are split up when additional values are used. The method is used whereby a text must be specified from additional value to additional value and then the used additional value with type information.
- The maximum length of a user-defined message is 160 characters. The user must ensure that the total length is not exceeded. If this happens, the last characters are truncated.
- An ARRAY OF structure is used for the area of user-defined messages.
- User-defined messages can contain up to four additional values.
  For this reason, messages that transfer additional values to the message handling, must be split up according to the following structure.

Table 4- 2 Structure for user-defined messages in STRING format *sLMsgHdlUserMessagesType*

Parameter	Data type	Description
sgLMsgHdlTextPart1	STRING[160]	First part of the message in STRING format up to additional value 1.
ab8LMsgHdlAdditionalValue1	sLMsgHdlAdditionalValueType	Number and format of the additional value to be used.
sgLMsgHdlTextPart2	STRING[90]	Second part of the message in STRING format from additional value 1 to additional value 2.

Parameter	Data type	Description
ab8LMsgHdlAdditionalValue2	sLMsgHdlAdditionalValueType	Number and format of the additional value to be used.
sgLMsgHdlTextPart3	STRING[50]	Third part of the message in STRING format from additional value 2 to additional value 3.
ab8LMsgHdlAdditionalValue3	sLMsgHdlAdditionalValueType	Number and format of the additional value to be used.
sgLMsgHdlTextPart4	STRING[50]	Fourth part of the message in STRING format from additional value 3 to additional value 4.
ab8LMsgHdlAdditionalValue4	sLMsgHdlAdditionalValueType	Number and format of the additional value to be used.

Table 4- 3 Structure for additional values *sLMsgHdlAdditionalValueType*

Parameter	Data type	Initial value	Description
b8ValueNumber	BYTE	1	The following values can be transferred here: 0: No additional value at this position (b8ValueType then has no significance) 1: additionalValue1 2: additionalValue2 3: functionBlockId 4: errorCode 5: additionalValueREAL
b8ValueType	BYTE	0	The following values can be transferred here: 0: Decimal display format \%d 1: Hexadecimal display format \%X 2: Floating-point display format \%lf

Example of a user-defined message with four additional values:

Table 4- 4 Extract from the **cPublic** program unit

```
//=====================================================================
//    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;
```

'User-defined message 8. FB/FC:/3/%d. AddInfo1:/1/%d, AddInfo2:/2/%d, ErrorCode:/4/%X'

This message must be entered as follows in the message handling:

Table 4- 5 Example

```
// Display of functionBlock Id
userDefinedMessages[7].sgLMsgHdlTextPart1 := 'User-defined message 8.
FB/FC:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue1.b8ValueNumber := 3;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue1.b8ValueType := 0;
// Display of additionalValue1
userDefinedMessages[7].sgLMsgHdlTextPart2 := '. Additional info 1:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue2.b8ValueNumber := 1;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue2.b8ValueType := 0;
// Display of additionalValue2
userDefinedMessages[7].sgLMsgHdlTextPart3 := ', Additional info 2:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue3.b8ValueNumber := 2;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue3.b8ValueType := 0;
// Display of errorCode
userDefinedMessages[7].sgLMsgHdlTextPart4 := ', Error code:';
userDefinedMessages[7].ab8LMsgHdlAdditionalValue4.b8ValueNumber := 4;
userDefinedMessages[7].ab8LMsgHdlAdditionalValue4.b8ValueType := 1;
```

If a user-defined message does not transfer any additional values, the message text is entered in the element *sgLMsgHdlTextPart1*. The other parts of the structure do not have to be filled. The sum of all the messages defined by the user must be entered in the LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS constant in the **cPublic** unit. The number of messages from the FBs/FCs is set in the LMSGHDL_NUMBER_OF_FUNCTION_BLOCK_IDS constant in the **cPublic** unit. The message texts that are to be generated for a message from FBs/FCs are part of the user-defined messages and must not be entered separately. The count of the user-defined messages must always start at 1 (one), up to the number of the respective type.

If a REAL value is to be output in one of the previous *additionalValues1* or *2*, this can be done as follows:

additionalValue1DINT := DWORD_TO_DINT(REAL_TO_DWORD(2.45286))

4.2.4 Embedding of the AlarmS handling or message bit handling

AlarmS handling

AlarmS can be activated by the user by setting the LMSGHDL_ALARM_S_USER_MESSAGES constant in the **cPublic** unit of the **LMsgHdl** library to TRUE.

When AlarmS is activated, the associated AlarmS message is issued for each user-defined message. This is performed within the call of the **FCLMsgHdlWriteUserMessageToBuffer** and **FCLMsgHdlWriteFBFCMessageToBuffer** functions for the transfer of user-defined messages.

It is not mandatory that each user-defined message be assigned an AlarmS message. Transfer of STRUCTALARMID#NIL in the *alarmSInfo* array in the **fMsgHdlInit** program unit

of the **FCLMsgHdlUserDefinedInfoForMessageHandling** function informs the message handling that an AlarmS message is not to be generated here. The system acknowledgement of the active AlarmS messages is performed by acknowledgement of all the active messages by the message handling.

The message configuration is performed in SIMOTION SCOUT. After selecting the project, click **Messages -> Configure** in the context menu (right-click). The **Message Configuration** window opens. After the user-defined AlarmS messages have been edited or imported into the project, they must be announced in the message handling. Detailed information on the message configuration in SIMOTION SCOUT can be found in the online help at the index entry *Message configuration*.

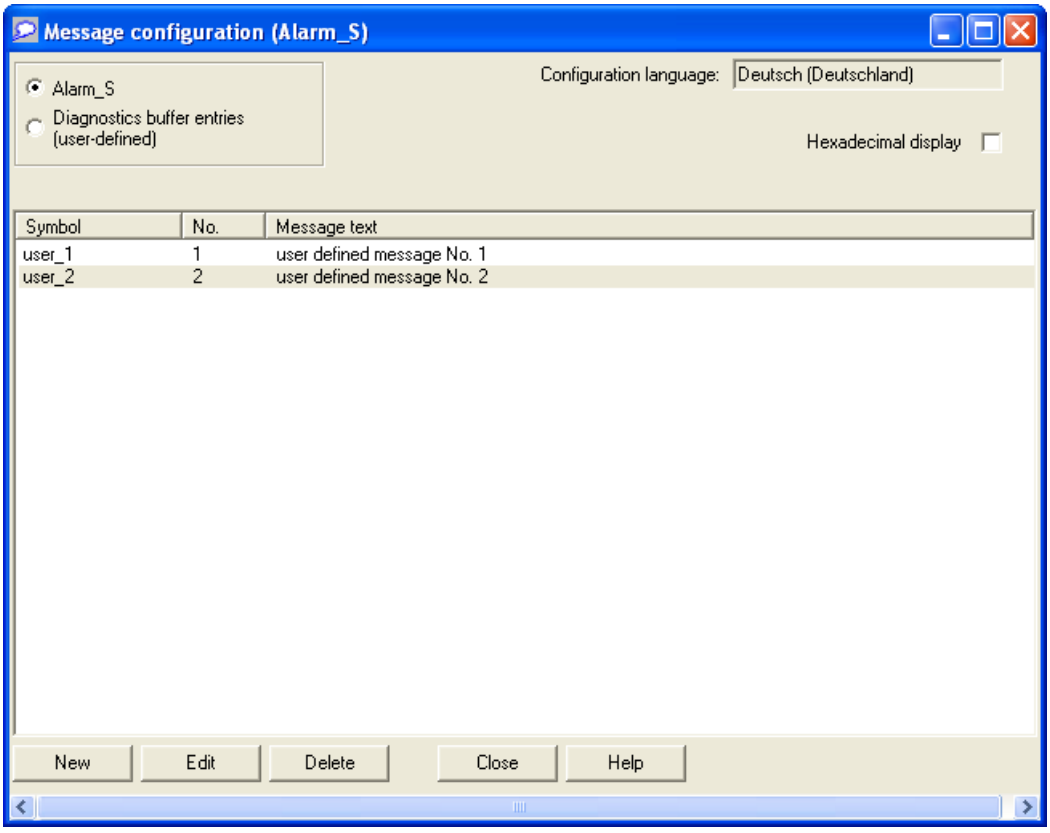

Figure 4-8 AlarmS messages (example)

The AlarmS messages are parameterized by the user in the **fLMsgHdlInit** program unit in the **FCLMsgHdlUserDefinedInfoForMessageHandling** function. The two messages in the above figure are transferred to the *gasLMsgHdlAlarmSInfo* global variable.

Table 4- 6 Extract from the **FCLMsgHdlUserDefinedInfoForMessageHandling** function

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

The symbol of the message and the type of message must be transferred by the user for each AlarmS message. The symbol must be specified with the name space for AlarmS

_Alarm following by a dot. The type of message specifies whether the alarm has to be acknowledged or not. AlarmSQ must be acknowledged by the HMI, AlarmS only by the system.

Note

When AlarmSQ messages are acknowledged by the message handling, the AlarmSQ messages are only acknowledged there. The display of these messages in the HMI must be acknowledged separately by the user.

The message issued with active AlarmS handling is determined by the event number (number of the user-defined message), which is transferred to the **FCLMsgHdlWriteUserMessageToBuffer** and **FCLMsgHdlWriteFBFCMessageToBuffer** functions. The event number corresponds to the subindex -1 in *gasLMsgHdlAlarmSInfo*. The AlarmS is therefore issued as follows:

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

This allows the appropriate AlarmS to be assigned to a certain user-defined message without both messages requiring the same number in the message handling. If a user-defined message is not to trigger an AlarmS, a value is not assigned to the relevant *sAlarmId* (STRUCTALARMID#NIL). Also note that each AlarmS message can only be active once. Therefore, an active AlarmS message can only be issued again after it has been acknowledged. The system acknowledgement of the active AlarmS messages is also performed by acknowledgement of all the active messages by the message handling. The SIMOTION SCOUT online help provides additional information on AlarmS.

Message bit handling

The message bit handling can be activated by setting the LMSGHDL_MESSAGE_BIT_USER_MESSAGES constant in the **cPublic** unit of the **LMsgHdl** library to TRUE. The active messages are transferred to the message bit handling in the *gab16LMsgHdlEventFlag* array from the **fLMsgHdl** program unit. If a user-defined message is set through the call of the **FCLMsgHdlWriteUserMessageToBuffer** and **FCLMsgHdlWriteFBFCMessageToBuffer** functions, the appropriate bit is also set in *gab16LMsgHdlEventFlag*. The set bit corresponds to message number -1 of the user-defined message. The message bits are set in an array of the WORD type. This results in the bit to be set from the message number:

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

The relevant bit for the issued message is set in the *gab16LMsgHdlEventFlag* variable. It is displayed, or transferred, whether the appropriate message bit has been or has to be acknowledged.

Note that when creating the messages, the index of the message always starts with 1 (one). A 0 (zero) is not permitted.

4.2.5 Defining machine error classes

Predefined machine error classes

Note

All entries or changes that have been made by the user are restored from the configuration when this has to be started again by the user.

The following machine error classes are already predefined in the **fLMsgHdlInit** program unit.

Table 4- 7 Predefined machine error classes in the **cPublic** unit

```
//=============================================================
//         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;
```

There are three different sources from which a machine error class can be set:

- Machine error class through user-defined messages
- Machine error class through peripheral fault messages
- Machine error class through user-defined FBs/FCs

These are described below.

Machine error class through user-defined messages

Each message is assigned an event number when a user-defined message is issued. The machine error class corresponding to an event number is entered in a table.

Table 4- 8 Assignment of the event number to the machine error class - example

Event number	Machine error class
1	Machine error class 4
2	Machine error class 3
3	Machine error class 1
4	Machine error class 1
5	Machine error class 2
...	...

This table is provided by the global array *gai8LMsgHdlUserDefinedMachineErrors* in the message handling and must be adapted by the user. The initialization of the user-defined message can be performed in the **fLMsgHdlInit** program unit in the

FCLMsgHdlUserDefinedInfoForMessageHandling function. Whereby the subindex in *userDefinedMachineErrors* corresponds to event number -1 of the appropriate user-defined message.

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

Machine error class through peripheral fault messages

Peripheral fault messages can also be assigned a machine error class. A machine error class is assigned to each peripheral device. A machine error class is only set when a negative message is present, e.g **station failure**, but not for a **station recovery**.

The machine error classes for peripheral fault messages are set in the **fLMsgHdlInit** program unit in the **FCLMsgHdlUserDefinedInfoForMessageHandling** function. The transfer of the machine error class for the failure of a peripheral fault message is performed in the *gasLMsgHdlPeripheralDevices* global variable. The message class is set here in the variable *peripheralDevices[0].i8MachineErrorClass*. The subindex belonging to the peripheral module for setting the machine error class can be found in the **fLMsgHdlInit** program unit in the **FCLMsgHdlInitProjectInfo** function. The information for the peripheral modules is automatically set by the script there.

Table 4- 9 Assignment of the peripheral device to the machine error class - example

Peripherals	Machine error class
Peripheral device 0	Machine error class 4
Peripheral device 1	Machine error class 3
Peripheral device 2	Machine error class 2

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

Machine error class through user-defined FBs/FCs

As errors with different severity can occur in a function block or function defined by the user, a corresponding machine error class can be set here depending on the error class. With the aid of a matrix, the machine error class is generated from the function block error classes.

Table 4- 10 Assignment of the FBs/FCs to the machine error classes (machine EC)

ID FB/FC	Error class 0	Error class 1	Error class 2	Error class 3
ID FB/FC 1	Machine EC 1	Machine EC 1	Machine EC 2	Machine EC 2
ID FB/FC 2	Machine EC 1	Machine EC 2	Machine EC 3	Machine EC 4
ID FB/FC 3	Machine EC 1	Machine EC 2	Machine EC 3	Machine EC 4
ID FB/FC 4	Machine EC 1	Machine EC 2	Machine EC 3	Machine EC 4
ID FB/FC 5	Machine EC 1	Machine EC 1	Machine EC 2	Machine EC 3

Each function block / function can trigger up to four different machine error classes, 0 to 3. The matrix can be used to parameterize how relevant an error is for the overall machine operation. In the above example, it can be seen that errors on FBs/FCs 1 and 5 trigger lower machine error classes as errors on the other FBs/FCs.

The configuration of this matrix is performed in the **fLMsgHdlInit** program unit in the **FCLMsgHdlUserDefinedInfoForMessageHandling** function. The matrix is transferred in the *gasLMsgHdlFBFCMachineErrorClasses* global variable.

```
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;
```

The currently active machine error classes are transferred to two variables by the message handling. These variables are declared in the **fLMsgHdl** program unit.

- *gi8LMsgHdlMachineErrorClass*: The currently highest machine error class that is active in the application is displayed here.
- *gb32LMsgHdlMachineErrorClasses*: All the currently active machine error classes are displayed with bit code here. As this variable is of the DWORD type, there can be only 32 machine error classes in the message handling (0 to 31). Each of these machine error classes is displayed via the appropriate bit in the variable.

4.3 Displaying messages via SIMOTION IT

Functionality

The active messages can be displayed via the SIMOTION IT page.

System requirements

- As of SIMOTION firmware V4.1 SP4
- As of Internet Explorer 6 or Mozilla Firefox 3.5

Displaying messages via SIMOTION IT

1. To start the connection to the SIMOTION device, enter the IP address in the address line of the browser.
2. Open the **User's Area** page.
3. Open the **Messages** page.

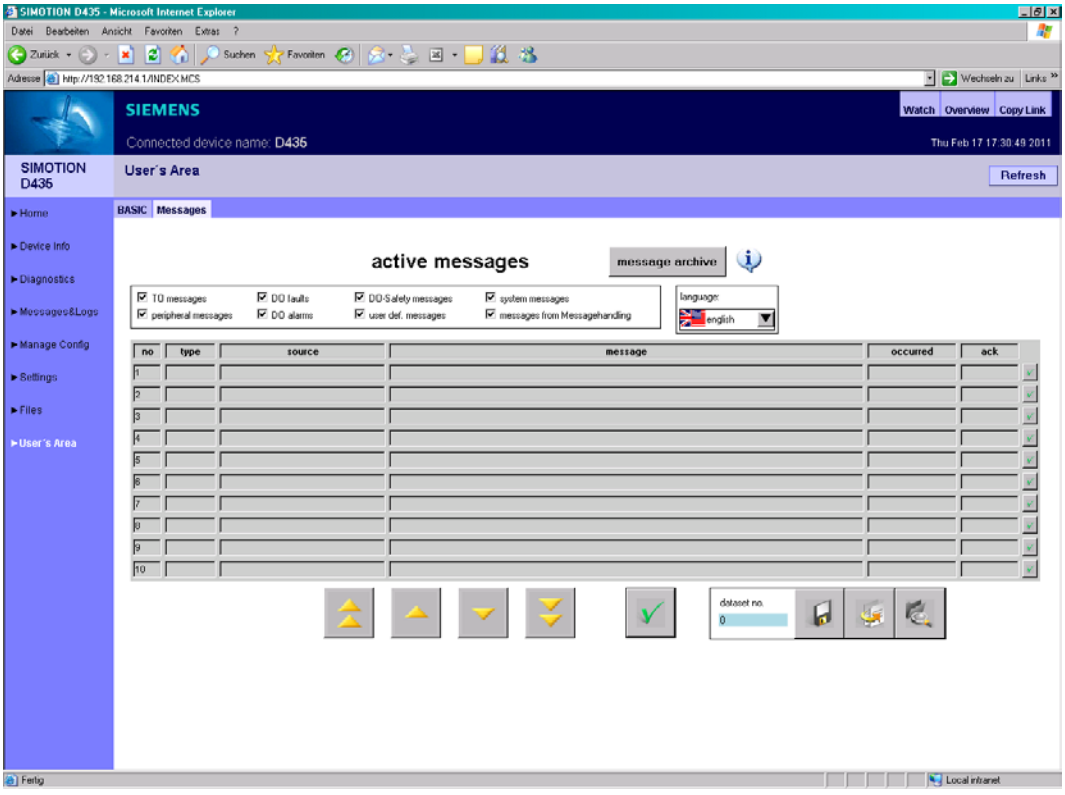

Figure 4-9 Message handling control panel start page

4.4 Important, frequently used variables

List of frequently used variables

The message handling is a complex application with numerous variables. As not all variables are always used by the user, the following table provides an overview of the most important variables.

Table 4- 11 Important, frequently used variables in the message handling

Unit	Variable	Meaning
fLMsgHdl	gb32LMsgHdlMachineErrorClasses	All the currently active machine error classes are displayed with bit code here. The maximum number is 32 (0 to 31).
	gi8LMsgHdlMachineErrorClass	Display of the active machine error class with the highest priority.
	gab16LMsgHdlEventFlag	Bits for message bit handling. Bit array for display of the active user-defined messages for the message bit handling.
	gsLMsgHdlActiveMessageTypes	Bit array for the message sources of active messages.
	gsLMsgHdlMessageLogString	Message log in STRING format.
pLMsgHdl	gboLMsgHdlInitDriveReady	Shows that the initialization software of the message handling has been run through in the BackgroundTask and that the message handling is active.
	gboLMsgHdlActivateNewMoMaData	With TRUE, the information transferred during runtime for modular machines is activated. After activation, the flag is removed by the message handling.
	gboLMsgHdlGlobalAcknowledge	Global acknowledgement of all active errors, with rising edge.
	gi32LMsgHdlNumberOfMessageInLog	Transfer of the number of the message to be acknowledged (only with single acknowledgement).
	gboLMsgHdlStartChangeLanguage	With TRUE, start of the language selection. Is automatically reset by the message handling.
	gu8LMsgHdlActiveLanguage	Transfer of the language to be loaded
	gboLMsgHdlStartWriteComplete MessageLogToStorageMedium	With TRUE, start of the storage of the current message log on the storage medium of the SIMOTION device. Is automatically reset by the message handling.
	gu32LMsgHdlDatasetNoForExport MessageLog	Name of the file in which the current message log is to be saved
	gboLMsgHdlUpdateHMI	Update of the displayed active messages.
	gboLMsgHdlUpdateHMILog	Update of the displayed active message log.

Unit	Variable	Meaning
	gboLMsgHdlScrollDown	Scroll down in the list of active messages with rising edge.
	gboLMsgHdlScrollDown1	Scroll down one line in the list of active messages with rising edge.
	gboLMsgHdlScrollUp	Scroll up.
	gboLMsgHdlScrollUp1	Scroll up one line in the list of active messages with rising edge.
	gboLMsgHdlGoToTop	Jump to the start of the active messages.
	gboLMsgHdlGoToEnd	Jump to the end of the active messages.
	gboLMsgHdlScrollDownLog	Scroll down in the message log.
	gboLMsgHdlScrollDown1Log	Scroll down one line.
	gboLMsgHdlScrollUpLog	Scroll up.
	gboLMsgHdlScrollUp1Log	Scroll up one line.
	gboLMsgHdlGoToTopLog	Jump to the start of the message log.
	gboLMsgHdlGoToEndLog	Jump to the end of the message history.
	gu8LMsgHdlScrollStep	Number of lines to scroll down or scroll up.
	gu8LMsgHdlNumberOfLinesForHMI	Number of messages that can be displayed on the HMI.
	gsLMsgHdlActiveMessageString	List of the active messages in STRING format.
	gsLMsgHdlActiveMsgToHMI	List of the active messages that are to be output on the HMI.
	gsLMsgHdlLogMsgToHMI	List of the message log that is to be output on the HMI.
fLMsgHdlInit	gasLMsgHdlUserDefinedMessages	Description of the user-defined messages.

Description of functions

5

5.1 General information on the description of functions

The following FBs and FCs relevant for the user are integrated into the message handling:

- **FBLMsgHdlActiveMsgSgToHMI** function block (fMsgHdl unit in LMsgHdl)
- **FBLMsgHdlMsgLogSgToHMI** function block (fMsgHdl unit in LMsgHdl)
- **FBLMsgHdlActiveMsgBaseDataToHMI** function block (fMsgHdl unit in LMsgHdl)
- **FBLMsgHdlMsgLogBaseDataToHMI** function block (fMsgHdl unit in LMsgHdl)
- **FCLMsgHdlWriteUserMessageToBuffer** function (fLMsgHdl unit)
- **FCLMsgHdlWriteFBFCMessageToBuffer** function (fLMsgHdl unit)

These are described in the following sections.

5.2 FBLMsgHdlActiveMsgSgToHMI function block

5.2.1 General information on the function block

Note

Only strings of length 80 can be processed in WinCC flexible. For this reason, the message texts from the active messages must be split into two substrings. However, the data length can be changed via the LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI constant in **cPublic** of the **LMsgHdl** library when another HMI is used.

The **FBLMsgHdlActiveMsgSgToHMI** function block is used to display message texts of the global buffer for active messages in STRING format on an HMI.

The function block completes its processing in a processing cycle of the task in which it is called. The FB should preferably be called in the BackgroundTask. It only responds to falling edges at the relevant inputs. If several inputs are set at a call, only the first function is executed. The logical sequence for evaluating the input signals is:

- updateHMI
- scrollDown1
- scrollDown
- scrollUp1
- scrollUp

- goToTop
- goToEnd

As only part of the entire buffer for active messages can be displayed on the HMI, the function block checks independently whether the appropriate end of the buffer for all active messages is reached or not when scrolling **up** or **down**. When an end is reached, further movement in the relevant direction has no effect in the function block.

The user can jump directly to the start of the end of the list via *goToTop* and *goToEnd*.

If, for example, scroll up ten messages is requested, but there are only three messages in the buffer up to the end, the list in the HMI is also only moved up three messages.

The LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI constant of the **cPublic** unit specifies the maximum number of messages that can be transferred to an HMI. The *numberOfLinesForHMI* input informs the function block how many messages it should actually provide at the output in the *activeMsgToHMI* structure. In this way, it is possible to supply different HMIs with different numbers of message texts via separate instances of the function block. The maximum number of lines in the LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI constant must not be exceeded. The number of lines to be scrolled when the *scrollUp* and *scrollDown* inputs are actuated is specified in the *numberOfLinesToScroll* input variable.

The active messages in STRING format are output on an HMI at the *activeMsgToHMI* output.

5.2.2 Schematic representation in LAD/FBD

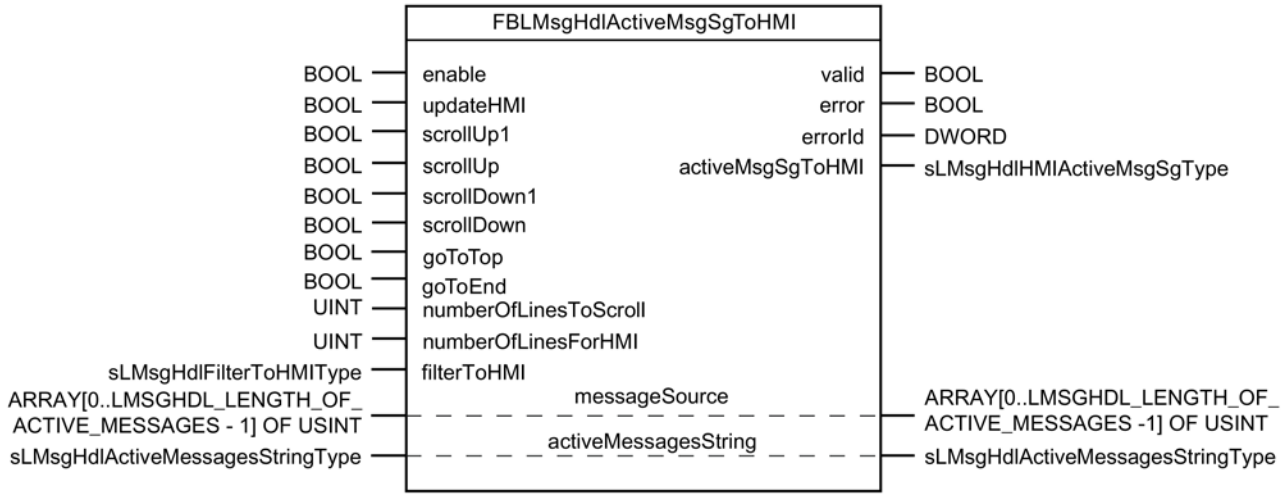

Figure 5-1 Schematic representation in LAD/FBD

5.2.3 Input and output parameters of the function block

The **FBLMsgHdlActiveMsgSgToHMI** function block has the following input and output parameters:

Table 5- 1 Input and output parameters

Name	Type [1]	Data type	M/O [2]	Initial value	Description
enable	IN	BOOL	M	FALSE	Function block enable.
updateHMI	IN	BOOL	O	FALSE	The currently output data area is updated on the HMI with a rising edge.
scrollUp1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up one message with a rising edge.
scrollUp	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up by the value transferred in the *numberOfLinesToScroll* variable with a rising edge.
scrollDown1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down one message with a rising edge.
scrollDown	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down by the value transferred in the *numberOfLinesToScroll* variable with a rising edge.
goToTop	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the top line on the latest entry with a rising edge.
goToEnd	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the bottom line on the oldest entry with a rising edge.
numberOf LinesToScroll	IN	UINT	O	1	Value that specifies how many lines the display area is to be moved when *scrollUp* or *scrollDown* is activated.
numberOf LinesForHMI	IN	UINT	O	1	Specifies the number of lines (messages) that are to be output for the HMI.
filterToHMI	IN	sLMsgHdl FilterToHMIType	M		Bit structure for the transfer of filter information for output on the HMI
messageSource	IN/OUT	ARRAY [0..LMSGHDL _LENGTH_OF_ ACTIVE_ MESSAGES - 1] OF USINT	M		The respective message source is transferred to the FB here for the filtering of the messages. This is generated automatically by the message handling in **pLMsgHdl-***gau8LMsgHdlActMessageStringMessageSource*.
Active MessagesString	IN/OUT	sLMsgHdl ActiveMessages StringType	M	-	Transfer of the current message buffer in STRING format.
valid	OUT	BOOL	-	FALSE	Display of the validity of the values at the outputs.
error	OUT	BOOL	-	FALSE	Displays whether an error has occurred while processing the FB.

Name	Type [1]	Data type	M/O [2]	Initial value	Description
errorId	OUT	DWORD	-	16#00000000	Returns the number of the error that has occurred.
activeMsgToHMI	OUT	sLMsgHdlHMI-ActiveMsg SgType	-	-	Return of the active messages in STRING format for display on an HMI.

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] Parameter type: M = mandatory parameter, O = optional parameter

Table 5- 2 Error messages

Error number [HEX]	Description
0	Error-free
9997	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum number of lines for the HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in **cPublic** unit)
9998	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum length of the transferred buffer (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES or LMSGHDL_LENGTH_OF_MESSAGE_LOG in **cPublic** unit)

5.2.4 Structure for parameter transfer

sLMsgHdlHMIActiveMsgType has the following structure.

Table 5- 3 Structure for active messages in STRING format on an HMI

Parameter	Data type	Initial value	Description
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array for information about which number the relevant message has in the buffer for active messages
asgMessageLevel	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[11]	Empty string	Array for information about the level of the message (fault, alarm, error, information)
asgMessageSource	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[64]	Empty string	Array for information about the source of the message
asgMessageText1	ARRAY[0..LMSGHDL_MAX_NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI]	Empty string	Array for language-dependent message text, section 1 (a message may only be maximum 160 characters long)

Parameter	Data type	Initial value	Description
asgMessageText2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Empty string	Array for language-dependent message text, section 2 (a message may only be maximum 160 characters long)
asgMessageOccured	ARRAY[0..LMSGHDL_NUMBER_OF_ LINES_MAX_FOR_HMI - 1] OF STRING[23]	Empty string	Time stamp when the relevant message occurred.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[17]	Empty string	Array for information about the type of acknowledgement for the message

5.3 FBLMsgHdlMsgLogSgToHMI function block

5.3.1 General information on the function block

Note

Only strings of length 80 can be processed in WinCC flexible. For this reason, the message texts from the active messages must be split into two substrings. However, the data length can be changed via the LMSGHDL_MAX_STRING_LENGTH_OF_MESSAGE_TEXTS_TO_HMI constant in **cPublic** of the **LMsgHdl** library when another HMI is used.

The **FBLMsgHdlMsgLogSgToHMI** function block is used to display message texts of the global buffer for the message log in STRING format on an HMI.

Response as described in FBLMsgHdlActiveMsgSgToHMI function block (Page 69).

5.3.2 Schematic representation in LAD/FBD

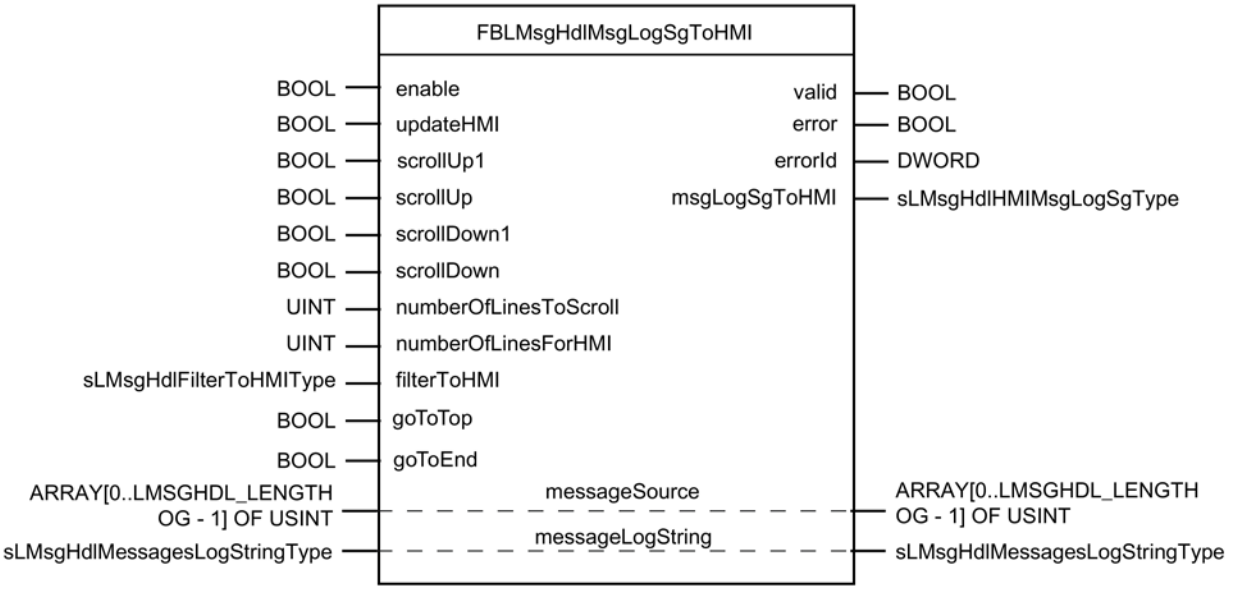

Figure 5-2 Schematic representation in LAD/FBD

5.3.3 Input and output parameters of the function block

The **FBLMsgHdlMsgLogSgToHMI** function block has the following input and output parameters:

Table 5- 4 Input and output parameters

Name	Type [1]	Data type	M/O [2]	Initial value	Description
enable	IN	BOOL	M	FALSE	Function block enable.
updateHMI	IN	BOOL	O	FALSE	The currently output data area is updated on the HMI with a rising edge.
scrollUp1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up one message with a rising edge.
scrollUp	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up by the value transferred in *numberOfLinesToScroll* with a rising edge.
scrollDown1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down one message with a rising edge.
scrollDown	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down by the value transferred in *numberOfLinesToScroll* with a rising edge.
goToTop	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the top line on the latest entry with a rising edge.

Name	Type [1]	Data type	M/O [2]	Initial value	Description
goToEnd	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the bottom line on the oldest entry with a rising edge.
numberOfLines-ToScroll	IN	UINT	O	1	Value that specifies how many lines the display area is moved when *scrollUp* or *scroll-Down* is activated.
numberOf LinesForHMI	IN	UINT	O	1	Specifies the number of lines (messages) that are to be output for the HMI.
filterToHMI	IN	sLMsgHdlFilter-ToHMIType	O		The information as to which message sources are to be displayed or not at the output of the FB is transferred here. If the use of SIMOTION IT is selected in the message handling, this information is provided in *gsLMsgHdlFilterToHMI* of the pLMsgHdl unit and should be transferred to the FB as VAR_IN_OUT.
messageSource	IN/OUT	ARRAY [0..LMSGHDL_ LENGTH_OF -1] OF UINT	M		The message sources belonging to the messages must be transferred to the FB here in USINT format. The message sources in USINT format are created by the message handling and are in *gau8LMsgHdlMessageLogStringMessageSource* of the pLMsgHdl unit. This variable must be transferred here as VAR_IN_OUT.
messageLog String	IN/OUT	sLMsgHdl MessageLog StringType	M	-	Transfers the message log buffer in STRING format.
valid	OUT	BOOL	-	FALSE	Displays the validity of the values at the outputs.
error	OUT	BOOL	-	FALSE	Displays whether an error has occurred while processing the FB.
errorId	OUT	DWORD	-	16#00000000	Returns the number of the error that has occurred.
msgLogToHMI	OUT	sLMsgHd lHMIMsg LogSgType	-	-	Returns the messages from the message log in STRING format for display on the HMI.

[1] Parameter types: IN = input parameters, OUT = output parameters, IN/OUT = in/out parameters

[2] Parameter type: M = mandatory parameter, O = optional parameter

Table 5- 5 Error messages

Error number [HEX]	Description
0	Error-free
9997	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum number of lines for the HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in **cPublic** unit)
9998	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum length of the transferred buffer (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES or LMSGHDL_LENGTH_OF_MESSAGE_LOG in **cPublic** unit)

5.3.4 Structure for parameter transfer

sLMsgHdlHMIMsgLogSgType has the following structure.

Table 5- 6 Structure for messages in the message log in STRING format on an HMI

Parameter	Data type	Initial value	Description
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array for information about which number the relevant message has in the buffer for the message log.
asgMessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[11]	Empty string	Array for information about the level of the message (fault, alarm, error, information).
asgMessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[64]	Empty string	Array for information about the source of the message.
asgMessageText1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX_STRING_ LENGTH_OF_MESSAGE_TEXTS_ TO_HMI]	Empty string	Array for language-dependent message text, section 1 (a message may only be maximum 160 characters long).
asgMessageText2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[LMDGHDL_MAX _STRING_LENGTH_OF_MESSAGE_ TEXTS_TO_HMI]	Empty string	Array for language-dependent message text, section 2 (a message may only be maximum 160 characters long).
asgMessageOccured	ARRAY[0..LMSGHDL_NUMBER_OF_ LINES_MAX_FOR_HMI - 1] OF STRING[23]	Empty string	Time stamp when the relevant message occurred.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[17]	Empty string	Array for information about the type of acknowledgement for the message.
asgMessageGone	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF STRING[23]	Empty string	Time stamp when the relevant message has gone.

For an overview of the user constants and structures, see Section Public constants (Page 22).

5.4 FBLMsgHdlActiveMsgBaseDataToHMI function block

5.4.1 General information on the function block

The **FBLMsgHdlActiveMsgBaseDataToHMI** function block is used to display messages of the global buffer for active messages in raw data format on an HMI.

Response as described in FBLMsgHdlActiveMsgSgToHMI function block (Page 69).

See also

Interpretation of the raw data (Page 115)

5.4.2 Schematic representation in LAD/FBD

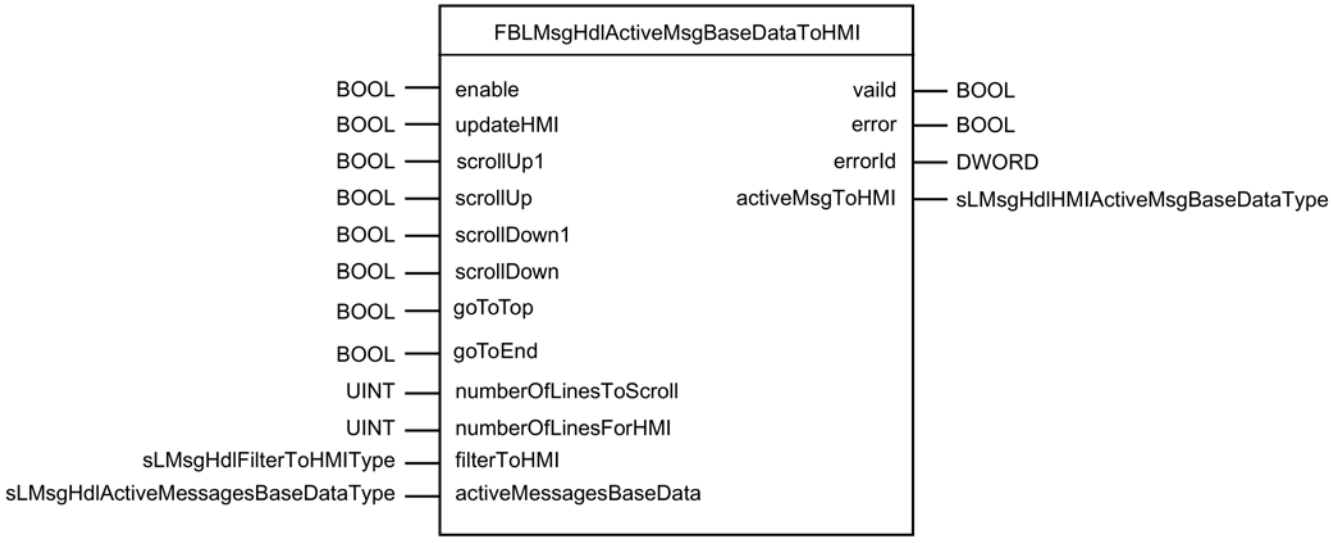

Figure 5-3 Schematic representation in LAD/FBD

5.4.3 Input and output parameters of the function block

The **FBLMsgHdlActiveMsgBaseDataToHMI** function block has the following input and output parameters:

Table 5- 7 Input and output parameters

Name	Type [1]	Data type	M/O [2]	Initial value	Description
enable	IN	BOOL	M	FALSE	Function block enable.
updateHMI	IN	BOOL	O	FALSE	The currently output data area is updated on the HMI with a rising edge.
scrollUp1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up one message with a rising edge.
scrollUp	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up by the value transferred in the *numberOfLinesToScroll* variable with a rising edge.
scrollDown1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down one message with a rising edge.
scrollDown	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down by the value transferred in the *numberOfLinesToScroll* variable with a rising edge.
goToTop	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the top line on the latest entry with a rising edge.
goToEnd	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the bottom line on the oldest entry with a rising edge.
numberOfLines-ToScroll	IN	UINT	O	1	Value that specifies how many lines the display area is to be moved when *scrollUp* or *scrollDown* is activated.
numberOfLines ForHMI	IN	UINT	O	1	Specifies the number of lines (messages) that are to be output for the HMI.
filterToHMI	IN	sLMsgHdlFilter-ToHMIType	O		The information as to which message sources are to be displayed or not at the output of the FB is transferred here. If the use of SIMOTION IT is selected in the message handling, this information is provided in *gsLMsgHdlFilterToHMI* of the pLMsgHdl unit and should be transferred to the FB as VAR_IN_OUT.
activeMessages BaseData	IN_OUT	sLMsgHdl Active MessagesBase DataType	M	-	Transfers the current message log in raw data format.
valid	OUT	BOOL	-	FALSE	Displays the validity of the values at the outputs.
error	OUT	BOOL	-	FALSE	Displays whether an error has occurred while processing the FB.

Name	Type [1]	Data type	M/O [2]	Initial value	Description
errorId	OUT	DWORD	-	16#00000000	Returns the number of the error that has occurred.
activeMsgToHMI	OUT	sLMsgHdlHMI-ActiveMsg BaseDataType	-	-	Returns the active messages in raw data format for display on an HMI.

[1] Parameter types: IN = input parameter, OUT = output parameter, IN_OUT = in/out parameter

[2] Parameter type: M = mandatory parameter, O = optional parameter

Table 5- 8 Error messages

Error number [HEX]	Description
0	Error-free
9997	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum number of lines for the HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in **cPublic** unit)
9998	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum length of the transferred buffer (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES or LMSGHDL_LENGTH_OF_MESSAGE_LOG in **cPublic** unit)

5.4.4 Structure for parameter transfer

sLMsgHdlHMIActiveMsgSgType has the following structure.

Table 5- 9 Structure for active messages in STRING format on an HMI

Parameter	Data type	Initial value	Description
adtMessageOccured	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DT	DT#0001 -01-01-0:0:0	Array for information about the active messages occurred time stamp
ab32Parameter3	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 3
ab32Parameter4	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 4
ab32Parameter5	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 5
ab32Parameter6	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 6

Parameter	Data type	Initial value	Description
ab32Parameter7	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 7
ab32Parameter8	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 8
ab32Parameter9	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 9
ab32Parameter10	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 10
ab32Parameter11	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 11
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array for information about which number the relevant message has in the message buffer for active messages
ai16Parameter2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF INT	0	Array for information in parameter 2
au8MessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the message source
au8MessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the message level
au8AcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the type of acknowledgement
au8ErrorClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the error class
au16Parameter1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF UINT	0	Array for information in parameter 1

5.5 FBLMsgHdlMsgLogBaseDataToHMI function block

5.5.1 General information on the function block

The **FBLMsgHdlMsgLogBaseDataToHMI** function block is used to display message texts of the global buffer for the message log in raw data format on an HMI.

Response as described in FBLMsgHdlActiveMsgToHMI function block (Page 69).

See also

Interpretation of the raw data (Page 115)

5.5.2 Schematic representation in LAD/FBD

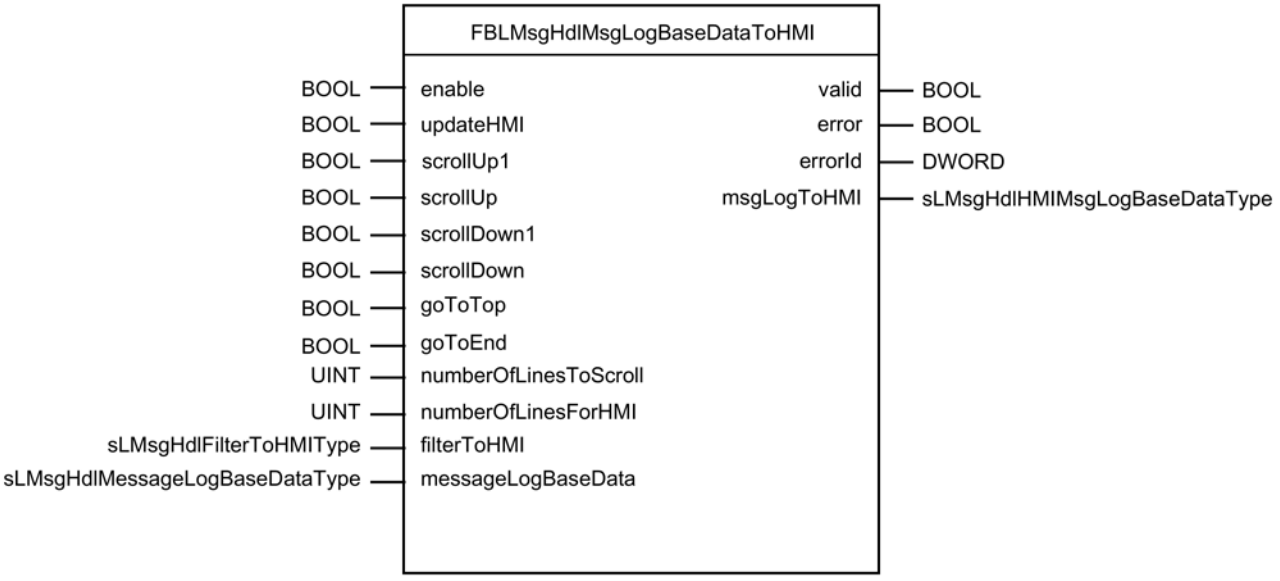

Figure 5-4 Schematic representation in LAD/FBD

5.5.3 Input and output parameters of the function block

The **FBLMsgHdlMsgLogBaseDataToHMI** function block has the following input and output parameters:

Table 5- 10 Input and output parameters

Name	Type [1]	Data type	M/O [2]	Initial value	Description
enable	IN	BOOL	M	FALSE	Function block enable.
updateHMI	IN	BOOL	O	FALSE	The currently output data area is updated on the HMI with a rising edge.
scrollUp1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up one message with a rising edge.
scrollUp	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved up by the value transferred in *numberOfLinesToScroll* with a rising edge.
scrollDown1	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down one message with a rising edge.
scrollDown	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved down by the value transferred in *numberOfLinesToScroll* with a rising edge.
goToTop	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the top line on the latest entry with a rising edge.
goToEnd	IN	BOOL	O	FALSE	The data area to be displayed on the HMI is moved with the bottom line on the oldest entry with a rising edge.
numberOfLines-ToScroll	IN	UINT	O	1	Value that specifies how many lines the display area is moved when *scrollUp* or *scrollDown* is activated.
numberOf LinesForHMI	IN	UINT	O	1	Specifies the number of lines (messages) that are to be output for the HMI.
filterToHMI	IN	sLMsgHdl FilterToHMIType	O		The information as to which message sources are to be displayed or not at the output of the FB is transferred here. If the use of SIMOTION IT is selected in the message handling, this information is provided in *gsLMsgHdlFilterToHMI* of the pLMsgHdl unit and should be transferred to the FB as VAR_IN_OUT.
Message LogBaseData	IN_OUT	sLMsgHdl MessageLog-BaseDataType	M	-	Transfers the current message log in raw data format.
valid	OUT	BOOL	-	FALSE	Displays the validity of the values at the outputs.
error	OUT	BOOL	-	FALSE	Displays whether an error has occurred while processing the FB.

Name	Type [1]	Data type	M/O [2]	Initial value	Description
errorId	OUT	DWORD	-	16#00000000	Returns the number of the error that has occurred.
msgLogToHMI	OUT	sLMsg HdlHMIMsgLog-gLog-BaseDataType	-	-	Returns the messages in raw data format for display on an HMI.

[1] Parameter types: IN = input parameter, OUT = output parameter, IN_OUT = in/out parameter

[2] Parameter type: M = mandatory parameter, O = optional parameter

Table 5- 11 Error messages

Error number [HEX]	Description
0	Error-free
9997	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum number of lines for the HMI (LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI in **cPublic** unit)
9998	The number of lines (messages) to be displayed on the HMI in the *numberOfLinesForHMI* parameter is greater than the maximum length of the transferred buffer (LMSGHDL_LENGTH_OF_ACTIVE_MESSAGES or LMSGHDL_LENGTH_OF_MESSAGE_LOG in **cPublic** unit)

5.5.4 Structure for parameter transfer

sLMsgHdlHMIMsgLogBaseDataType has the following structure.

Table 5- 12 Structure for the message log in STRING format on an HMI

Parameter	Data type	Initial value	Description
ai16NumberOfMessage	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_LINES_FOR_HMI - 1] OF DINT	0	Array for information about which number the relevant message has in the message buffer for active messages
au8MessageSource	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the message source
au8MessageLevel	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the message level
au8AcknowledgeClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the type of acknowledgement
au8ErrorClass	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF USINT	0	Array for information about the error class

Parameter	Data type	Initial value	Description
au16Parameter1	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF UINT	0	Array for information in parameter 1
ai16Parameter2	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF INT	0	Array for information in parameter 2
ab32Parameter3	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 3
ab32Parameter4	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 4
ab32Parameter5	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 5
ab32Parameter6	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 6
ab32Parameter7	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 7
ab32Parameter8	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 8
ab32Parameter9	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 9
ab32Parameter10	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information in parameter 10
ab32Parameter11	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DWORD	16#0000 0000	Array for information 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 for information about the active messages occurred time stamp
adtMessageGone	ARRAY[0..LMSGHDL_MAX_ NUMBER_OF_VISIBLE_LINES_FOR _HMI -1] OF DT	DT#0001 -01-01- 0:0:0	Array for information about message gone

5.6 FCLMsgHdlWriteUserMessageToBuffer and FCLMsgHdlWriteFBFCMessageToBuffer functions

5.6.1 General information on the functions

User-defined messages are transferred to the message handling when the **FCLMsgHdlWriteUserMessageToBuffer** or **FCLMsgHdlWriteFBFCMessageToBuffer** function is called. These messages are generated from the application. This function must be called once for each new message. For messages that are already active or have not been acknowledged yet, no new message is entered in the buffer, or a message is issued via the message bit handling or AlarmS handling.

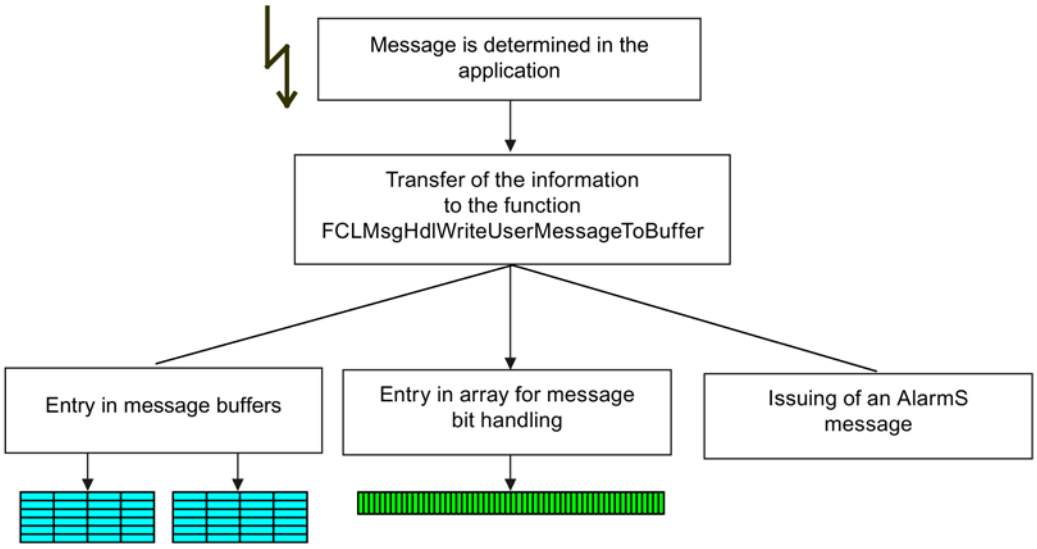

Figure 5-5 Entry of user-defined messages

A message is specified via its unique message number. The entered messages can be processed in three different ways for forwarding to the operator.

- Entry in the message buffer:
  The message is entered in the message buffers. When using the string-based buffers, the user must ensure that a message text is stored in the controller, see Section Integration of the application into a SIMOTION project (Page 51)
- Use of the message bit handling:
  When the message bit handling is selected by setting the LMSGHDL_MESSAGE_BIT_USER_MESSAGES constant to TRUE, the bit corresponding to the message number in the *gab16LMsgHldEventFlag* global WORD array is set in the **fLMsgHdl** unit. The message texts are stored in the HMI.
- Use of AlarmS:
  When the AlarmS message procedure is selected by setting the LMSGHDL_ALARM_S_USER_MESSAGES constant to TRUE, an AlarmS message corresponding to the bit number is generated. The message texts must be entered by the user in SIMOTION SCOUT.

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

The *eventNumber* determines the user-defined message that is output in the message handling. At the same time, the appropriate message is set when the AlarmS handling or message bit handling is active. If an additional *functionBlockId* is not set, the message class belonging to the event number is issued.

The *functionBlockId* is set if the user-defined message is from a function or function block. In this case, the message class is not generated from the event number, but from the appropriate specification of the *gasLMsgHdlFBFCMachineErrorClasses* variable. The value for the machine error class is read out as follows and taken into the message handling:
`gasLMsgHdlFBFCMachineErrorClasses[0].ai8ErrorClass[2]`

5.6.2 Schematic representation in LAD/FBD

FCLMsgHdlWriteUserMessageToBuffer function

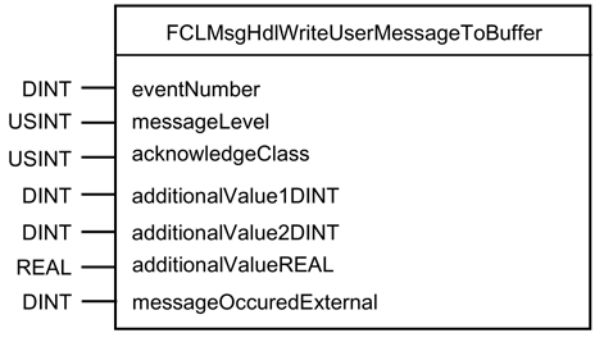

Figure 5-6 Schematic representation in LAD/FBD

FCLMsgHdlWriteFBFCMessageToBuffer function

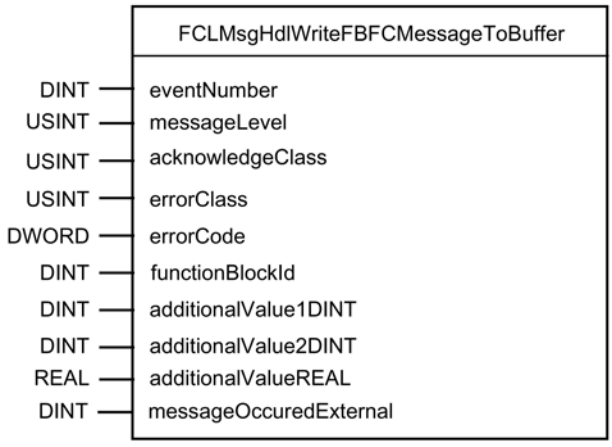

Figure 5-7 Schematic representation in LAD/FBD

5.6.3 Input and output parameters of the functions

The user must start with 1 (one) for the message numbers, 0 (zero) is not permitted.

Input and output parameters of the FCLMsgHdlWriteUserMessageToBuffer function

The **FCLMsgHdlWriteUserMessageToBuffer** function has the following input and output parameters:

Table 5- 13 Input and output parameters

Element	P type [1]	Data type	M/O [2]	Initial value	Meaning
eventNumber	IN	DINT	M	0	Message number
messageLevel	IN	USINT	O	2	Transfer of the message level of the message (default is 2 -> error)
acknowledgeClass	IN	USINT	O	2	Transfer of the acknowledgement type (default is 2 -> direct)
additionalValue1DINT	IN	DINT	O	0	Additional value for the error in DINT format, is transferred to AlarmS
additionalValue2DINT	IN	DINT	O	0	Additional value for the error in DINT format
additionalValueREAL	IN	REAL	O	0.0	Additional value to transfer errors in REAL format

Element	P type [1]	Data type	M/O [2]	Initial value	Meaning
messageOccuredExternal	IN	DINT	O	0	If this input is used, the value transferred there is transferred to the message handling. If no value is transferred via this input, the system time of the SIMOTION device applies. Default setting is the system time of the SIMOTION device.
FCLMsgHdlWriteMessage ToBuffer	OUT	VOID	-	-	No return code for the function

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter type: M = mandatory parameter, O = optional parameter

Input and output parameters of the FCLMsgHdlWriteFBFCMessageToBuffer function

The **FCLMsgHdlWriteFBFCMessageToBuffer** function has the following input and output parameters:

Table 5- 14 Input and output parameters

Element	P type [1]	Data type	M/O [2]	Initial value	Meaning
eventNumber	IN	DINT	M	0	Message number
messageLevel	IN	USINT	O	2	Transfer of the message level of the message (default is 2 -> error)
acknowledgeClass	IN	USINT	O	2	Transfer of the acknowledgement type (default is 2 -> direct)
errorClass	IN	USINT	O	0	Error class of the message
errorCode	IN	DWORD	O	16#00000000	Error code (e.g. for error response from function blocks), is transferred to AlarmS
functionBlockId	IN	DINT	O	0	ID of the function block triggering the message
additionalValue1DINT	IN	DINT	O	0	Additional value for the error in DINT format
additionalValue2DINT	IN	DINT	O	0	Additional value for the error in DINT format
additionalValueREAL	IN	REAL	O	0.0	Additional value to transfer errors in REAL format
messageOccuredExternal	IN	DINT	O	0	If this input is used, the value transferred there is transferred to the message handling. If no value is transferred via this input, the system time of the SIMOTION device applies. Default setting is the system time of the SIMOTION device.
FCLMsgHdlWriteFCFB MessageToBuffer	OUT	VOID	-	-	No return code for the function

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter type: M = mandatory parameter, O = optional parameter

5.7 Structure for message log as raw data

Table 5- 15 Structure for message log as raw data *sLMsgHdlMessageLogBaseDataType*

Parameter	Data type	Initial value	Description
i16ActualIndex	INT	0	Current index in the global message log for raw data
i16NumberOfNew Messages	INT	0	Only for internal use by message handling
boChangesInLog-BaseData	BOOL	FALSE	Only for internal use by message handling
boNewMessages ForHMILogBase	BOOL	FALSE	Only for internal use by message handling
boBuildNewString Messages	BOOL	FALSE	Only for internal use by message handling
au8MessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF USINT	0	Array for information about the source of the message
au8MessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF USINT	0	Array for information about the level of the message (fault, alarm, error, information)
au8AcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF USINT	0	Array for information about the type of acknowledgement for the message
au8ErrorClass	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF USINT	0	Array for information about the error class of a message
au16Parameter1	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF UINT	0	Array for information about Variable1 of the respective message type
ai16Parameter2	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF INT	0	Array for information about Variable2 of the respective message type
ab32Parameter3	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable3 of the respective message type
ab32Parameter4	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable4 of the respective message type
ab32Parameter5	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable5 of the respective message type
ab32Parameter6	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable6 of the respective message type
ab32Parameter7	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable7 of the respective message type
ab32Parameter8	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable8 of the respective message type

Parameter	Data type	Initial value	Description
ab32Parameter9	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable9 of the respective message type
ab32Parameter10	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable10 of the respective message type
ab32Parameter11	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DWORD	16#00000000	Array for information about Variable11 of the respective message type
adtMessageOccured	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DT	DT#0001-01-01-0:0:0:0	Time stamp when the relevant message occurred.
adtMessageGone	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF DT	DT#0001-01-01-0:0:0:0	Time stamp when the relevant message has gone.
boNewMessage	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] BOOL	FALSE	Shows whether the relevant message is a new entry in the raw data message log.
i16MessageIndexLastCycle	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] INT	0	Index of the message in the previous background cycle. Auxiliary variable for update of the message log in STRING format.

5.8 Structure for message log in STRING format

Table 5- 16 Structure for message log in STRING format *sLMsgHdlMessageLogStringType*

Parameter	Data type	Initial value	Description
i16ActualIndex	INT	0	Current index in the global message log in STRING format
boChangesInLogStringData	BOOL	FALSE	Only for internal use by message handling
boChangesInLogStringDataForHMI	BOOL	FALSE	Only for internal use by message handling
asgMessageLevel	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[11]	Empty string	Array for information about the level of the message (fault, alarm, error, information).
asgMessageSource	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[64]	Empty string	Array for information about the source of the message.
asgMessageText	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[160]	Empty string	Array for language-dependent message text.
asgMessageOccured	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[23]	Empty string	Time stamp when the relevant message occurred.
asgAcknowledgeClass	ARRAY[0..LMSGHDL_LENGTH_OF_ MESSAGE_LOG - 1] OF STRING[17]	Empty string	Array for information about the type of acknowledgement for the message.
asgMessageGone	ARRAY[0.LMSGHDL_LENGTH_OF_ MESSAGE_LOG- 1] OF STRING[23]	Empty string	Time stamp when the relevant message has gone.

Alarm and error messages

6

6.1 General information on the error handling

Errors can also occur during processing in the message handling. These are signaled when errors occur during the processing in the message handling, e.g. data exchange failure with drive objects of a SINAMICS drive.

6.2 Buffer overflow

The messages of the individual, possible message sources are first collected in buffers. This guarantees the data consistency of the message information. These buffers are configured as ring buffers and only release memory areas written by a new message when this information has been entered in the global message buffers.

If several new messages occur at a message source in a short time, it may happen that a buffer is written quicker than it can be read out. If this happens, a buffer may overflow. The overflow of one of these buffers is signaled by an internal error message to the message handling.

If such a message is entered, there is no longer any guarantee that the information in the message buffers is complete. All messages from a source that occur after an overflow are not taken into the message buffers and are lost. Only after there has been an acknowledgement via the message handling are new messages recognized again at the source.

The relevant buffer can be increased via the respective constant in **cPublic** of the **LMsgHdl** library. This value only takes effect however after recompilation and download of the project.

6.3 Overflow of AlarmS messages

When using AlarmS in the message handling, make sure that only 40 AlarmS messages can be active simultaneously. If the next message is issued, this can no longer be processed by the AlarmS handler and is lost. Despite this, the messages are still taken into the message handling. At the same time, the internal error **LMSGHDL_ALARM_S_ERROR (100002)** is triggered and entered in the message handling.

6.4 Error during startup

The message handling monitors the startup of the machine and collects generic information that is required later in the message handling. If all the configured modules are not available on the controller, there is no guarantee that the message handling functions correctly. For this reason, an internal message is signaled by the message handling as soon as a configured module is not available or a different operating state as active is signaled **LMSGHDL_ERROR_CHECK_STARTUP_OK (100026)**.

The timeout expires when acyclic data exchange to the SINAMICS_Integrated is not possible for a SIMOTION D.

After a station recovery message of an I/O module, the check as to whether the I/O module is available again is started automatically.

If an error occurs during startup, the startup check is immediately terminated and the message about the buffer management error transferred to the message handling. If the buffer management signals an error, the message handling can only be used again after a restart.

6.5 Messages by I/O modules

In the *gasLMsgHdlPeripheralDevices* variable of the *dProtected* library unit, each I/O module has a *boStationConnected* variable of the BOOL type which indicates whether the I/O module is available or not. The message handling sets this bit at a station failure or recovery.

6.6 DO safety messages

Safety messages are handled like errors on the drive object. Safety messages cannot be acknowledged by the message handling. For this reason, the **Message gone** time stamp is monitored. This means that each safety message when it occurs and when it is acknowledged creates an entry in the buffer for safety messages on all of the drive objects. In the message handling, make sure that active safety messages are automatically removed from the buffer for active messages when acknowledged. In addition, in the global **message log**, only the **Message gone** time stamp has to be added to an active safety message when it disappears.

The buffering for safety messages from the previous detection is in the *gasLMsgHdlAuxiliaryBufferDOWithTOSafety* variable. The newly found safety messages are stored in the *asLMsgHdlAuxiliaryBufferDOSafety* array.

6.7 User-defined messages

In the message handling, it is possible to transfer an independently generated time stamp when the message occurred for all user-defined messages. For this purpose, the *messageOccuredExternal* input of the DT type is available on all FCs. If a value other than the default value is transferred here, then this time stamp is transferred to the message handling instead of the current system time. If this input is not filled or written with a default value, the message handling takes over the current system time stamp.

It is possible to create the user-defined message with the *EventID = 0* variable. When this variable is assigned as user-defined message, it is possible to create a message several times. All other messages can only be active once. If the *EventId = 0* is assigned, the message is entered again in the message handling with the respective time stamp with each new call. However, this message does not generate a message class or an associated AlarmS and no bit in the message bit handling. The message text required for the STRING output is available in the **fLMsgHdlInit** unit in the *gasLMsgHdlUserDefinedMessageEvent0* variable. The user-defined message with *EventId = 0* can be created via the **FCLMsgHdlWriteFBFCMessageToBuffer** function. It is thus possible to transfer this message as additional values of an error code, a *functionBlockId* and two additional values. The user-defined messages generated in this way can also be acknowledged individually.

6.8 Error during data exchange with DOs

If an error occurs during data exchange with at least one DO, an appropriate message is transferred to the message handling. Data exchange with the relevant DO is then interrupted. Data exchange with the DO is only restarted after a global acknowledgement by the message handling. If the error in the data exchange occurs again, another message is output.

The DO that causes the message, is transferred to the message handling either via the number of the associated axis, the logical address or a logical address including the DO number, depending on the type of DO.

6.9 Particularity for alarms on drive objects

Alarms on drive objects on SINAMICS modules cannot be acknowledged. When these alarms occur, they remain present until the reason for the alarm no longer exists. As soon as an active alarm is no longer present, it is automatically deleted from the message buffers for active messages.

6.10 Particularity for peripheral messages

Messages through I/O modules have a different character in SIMOTION. There are **negative** and **positive** peripheral messages. For example, the message *Station failure* is a negative message. Whereas *Station recovery* is a positive message. The collection of these peripheral messages is therefore implemented as follows in the message handling.

If a negative peripheral message occurs on a machine, this is entered in the message log and the message buffer for active messages. Positive messages are only entered in the message log. At the same time, a search is made in the message buffer for active messages to see whether a negative message belonging to the positive message is still active. If this is the case, the appropriate message is acknowledged automatically and removed from the active messages. If a negative active peripheral message is acknowledged before the associated positive message occurs, it is also removed from the buffer for active messages. The occurrence of all peripheral messages can therefore always be tracked in the log memory of the message handling.

Table 6- 1 Associated positive and negative peripheral messages

Negative message	Positive message
ID 202 Station failure	ID 203 Station recovery
ID 204 Error when generating the process image	ID 206 Generation of the process image functions again
ID 210 Multiple clock failure or PLL unlocked	ID209 PLL locked in controlled operation
ID 215 Synchronization failed	ID214 Synchronization reached

The peripheral messages with the interrupt IDs 200, 201, 205, 208, 211, 212, 213, 216 and 217 are also taken into the log memory and the message buffer for active messages. However, these IDs have no complementary peripheral message and are not acknowledged automatically. The meanings of the IDs listed here are described in the SIMOTION SCOUT online help at **Using task start information**.

6.11 Reaction to internal errors

If an error occurs in the message handling, this is entered in the respective message buffers via an appropriate message. A global machine error class is also set. This machine error class is identical for all messages of the message handling and can be specified by the user in the **cPublic** unit of the **LMsgHdl** library via the LMSGHDL_MACHINE_ERROR_CLASS_ERROR_IN_MESSAGEHANDLING constant.

The messages generated by internal errors can be reset via the global acknowledgement. However, if the reason for the internal error has not been corrected at the time of the acknowledgement, the message is entered in the message handling again. As the internal errors in the message handling are application-specific messages, they are transferred to the message handling in the same way as user-defined messages. To clearly distinguish these messages from the user messages, the event numbers start at 100.000.

The following internal errors can occur in the message handling:

Table 6- 2 Internal error in the message handling

Name of the constant	Event number	Meaning
LMSGHDL_UNKNOWN_USER_EVENT		
	100.000	An unknown user-defined message has been transferred to the message handling. The faulty message can be identified by the additional values. This message can only be acknowledged.
LMSGHDL_EVENT_IN_UNKNOWN_TASK		
	100.001	A user-defined message has been transferred to the message handling from an unknown task. The execution system has been changed without recalling the script. The configuration script should be run through again.
LMSGHDL_ALARM_S_ERROR		
	100.002	An error occurred while issuing an AlarmS message. The internally used *_alarmSQId* and *_alarmSId* functions have generated an error. Remedy: See additional value *errorId*. (Errors of the system functions). This message can be acknowledged immediately.
LMSGHDL_OVERFLOW_BUFFER_EXECUTIONTASK_MESSAGES		
	100.003	The buffer to collect the ExecutionTask messages has overflowed. ExecutionFault messages have occurred in two successive starts of the machine, without the message handling being able to process the first message after the restart. Search for the error and restart the machine. Triggering message could not be taken into the buffers.
LMSGHDL_OVERFLOW_BUFFER_APPLICATION_MESSAGES		
	100.004	The buffer to collect the user messages has overflowed. The constant LMSGHDL_NUMBER_OF_APPLICATION_MESSAGES must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be taken into the buffers.
LMSGHDL_OVERFLOW_BUFFER_TECHFAULT_MESSAGES		
	100.005	The buffer to collect the TechnologicalFaultTask messages has overflowed. The constant LMSGHDL_NUMBER_OF_PERIPHERAL_FAULT_MESSAGES must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be taken into the buffers.
LMSGHDL_OVERFLOW_BUFFER_PERIPHERAL_MESSAGES		
	100.006	The buffer to collect the PeripheralFaultTask messages has overflowed. The LMSGHDL_NUMBER_OF_TECH_FAULT_MESSAGES constant must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be inserted into the buffers.
LMSGHDL_OVERFLOW_BUFFER_TIMEFAULT_MESSAGES		
	100.007	The buffer to collect the TimeFaultTask messages has overflowed, e.g. due to an endless loop in the BackgroundTask. The LMSGHDL_NUMBER_OF_TIME_FAULT_MESSAGES constant must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be taken into the buffers.
LMSGHDL_OVERFLOW_BUFFER_DOFAULT_MESSAGES		
	100.008	The buffer to collect the fault messages on all drive objects has overflowed. The LMSGHDL_NUMBER_OF_DO_FAULT_MESSAGES constant must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be taken into the buffers.

Name of the constant	Event number	Meaning
LMSGHDL_OVERFLOW_BUFFER_DOALARM_MESSAGES		
	100.009	The buffer to collect the alarm messages on all drive objects has overflowed. The LMSGHDL_NUMBER_OF_DO_ALARM_MESSAGES constant must be increased in **cPublic**. An acknowledgement is possible. Triggering message could not be taken into the buffers.
LMSGHDL_ERROR_IN_BUFFER_MANAGER		
	100.010	An error occurred while using the buffer management. As the buffer management for DP-V1 services has an error, the SINAMICS drive objects are no longer monitored. A machine restart is required.
LMSGHDL_ERROR_IN_CHECK_STARTUP		
	100.011	An error has occurred for the startup check in the **FBLDPV1CheckStartup** function block. Not all the configured I/O modules are available. The message handling cannot provide all the required information. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_ADD_TO		
	100.012	An attempt was made to transfer a non-existing axis to the message handling. The configuration script must be run through again.
LMSGHDL_ERROR_SEARCH_ALL_DOS		
	100.013	An error has occurred in the internal assignment of technology objects to drive objects, caused by the **FBLDPV1SearchAllDo** function. See transfer parameter *errorId* and DO-TO documentation in LDPV1. The message handling cannot provide all the required information. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_CHECK_WRITE_ACCESS		
	100.014	An error has occurred while checking whether the parameters can be written in the SINAMICS drive objects. See transfer parameter *errorId* and *CheckStartup* documentation in LDPV1. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_TIME_SYNC		
	100.015	An error occurred during the time synchronization of a SINAMICS module. See transfer parameter *errorId* and *TimeSync* documentation in LDPV1. Messages and drive objects may only have the time stamp of the SIMOTION RTC and cannot be assigned correctly. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_AT_AXIS		
	100.016	An error has occurred while determining faults/alarms on a drive object that belongs to a TO axis. See transfer parameter *errorId* and *GetFault* documentation in LDPV1. A SINAMICS module might have failed completely. The machine must be checked. An immediate acknowledgement is possible.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_CYCLIC_DOS		
	100.017	An error has occurred while determining faults/alarms on a drive object with cyclic data exchange. See transfer parameter *errorId* and *GetFault* documentation in LDPV1. A SINAMICS module might have failed completely. The machine must be checked. An immediate acknowledgement is possible.
LMSGHDL_ERROR_FB_GET_DO_MESSAGES_ACYCLIC_DOS		
	100.018	An error has occurred while determining faults/alarms on a drive object without cyclic data exchange. See transfer parameter *errorId* and *GetFault* documentation in LDPV1. A SINAMICS module might have failed completely. The machine must be checked. An immediate acknowledgement is possible.
LMSGHDL_ERROR_MESSAGE_BUFFER_MANAGER		

Name of the constant	Event number	Meaning
	100.019	An error has occurred in the block to create the message buffer (acknowledgement of the active messages). A SINAMICS module might have failed completely. The machine must be checked. An immediate acknowledgement is possible.
LMSGHDL_ERROR_GET_DO_NAME		
	100.020	Error occurred during automatic determination of the names of all configured drive objects. The message handling cannot provide all the required information. A SINAMICS module might have failed completely. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_INIT_MESSAGELOG_STRING		
	100.021	Error occurred while creating the message information in the STRING format. The texts stored in the system in STRING format must be checked.
LMSGHDL_ERROR_UPDATE_HMI_ACTIVE_MESSAGES		
	100.022	Error in the block for the output of the active messages in STRING format to the HMI or SIMOTION IT. A check of the transfer parameters for the output to an HMI is necessary. An immediate acknowledgement is possible.
LMSGHDL_ERROR_UPDATE_HMI_MESSAGE_LOG		
	100.023	Error in the block for the output of the message log in STRING format to the HMI or SIMOTION IT. A check of the transfer parameters for the output to an HMI is necessary. An immediate acknowledgement is possible.
LMSGHDL_ERROR_CHANGE_LANGUAGE_SYSTEM		
	100.024	The file of the language selected for system messages is not available on the storage medium of the SIMOTION device or is faulty. An immediate acknowledgement is possible.
LMSGHDL_ERROR_CHANGE_LANGUAGE_USER		
	100.025	The file of the language selected for user-defined messages is not available on the storage medium of the SIMOTION device or is faulty. An immediate acknowledgement is possible.
LMSGHDL_ERROR_CHECK_STARTUP_OK		
	100.026	Not all the configured devices have been identified as ready. The message handling cannot provide all the required information. An I/O module might have failed completely. The machine must be checked and, if required, restarted.
LMSGHDL_ERROR_CHANGE_LANGUAGE		
	100.027	Error occurred while changing the language for output of the messages in STRING format. It is not necessary to check the language files on the storage medium. An immediate acknowledgement is possible.
LMSGHDL_ERROR_WRITE_MESSAGE_LOG_TO_STORAGE_MEDIUM		
	100.028	Error occurred while writing to the message log on the storage medium of the SIMOTION device. There may not be sufficient system resources available. An immediate acknowledgement is possible.
LMSGHDL_OVERFLOW_BUFFER_DOSAFETY_MESSAGES		
	100.029	The buffer to collect the safety messages on all drive objects has overflowed. The LMSGHDL_NUMBER_OF_DO_SAFETY_MESSAGES constant must be increased in the **cPublic** unit. An acknowledgement is possible. The triggering message could not be taken over.
LMSGHDL_ILLEGAL_FUNCTION_BLOCK_ID		
	100.030	An illegal *functionBlockId* was transferred when a user-defined message was called by FBs/FCs. An immediate acknowledgement is possible.

Name of the constant	Event number	Meaning
LMSGHDL_ILLEGAL_ERROR_CLASS		
	100.031	An illegal *errorClass* was transferred when a user-defined message was called by FBs/FCs. An immediate acknowledgement is possible.
LMSGHDL_ERROR_IN_PERSITENT_DATA_POWER_MONITORING		
	100.032	An error has occurred with the voltage for the backup of the non-volatile data.

Application example 7

7.1 Defining machine error classes (example)

This application example shows the most important adaptations of a SIMOTION project. The programming of the messages depends on the requirements of the user. This example can therefore only show ideas for the implementation.

The following has been defined in this application example:

- Ten machine error classes
- Ten user-defined messages
- Messages for three FBs/FCs, each with four error classes
- Messages for two peripheral devices (SINAMICS Integrated drive and CU310 control unit)

Editing machine error classes

The required error classes must be edited in the **fLMsgHdlInit** program unit.

Editing machine error classes for user-defined messages

Create the machine error classes as global constants. You can define a maximum of 31 machine error classes. Machine error classes can be defined for user-defined messages and messages for peripheral devices.

Table 7- 1 Global constants

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

    //================================================================
    //         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
```

Comment out the initialization of the machine error classes and adapt the machine error classes to the structure that you require.

Table 7- 2 Initializing machine error classes for user-defined messages

```
//=====================================================================
// 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;
```

Assigning machine error classes for messages from FBs/FCs

Table 7- 3 Initializing machine error classes for messages from FBs/FCs

```
//=====================================================================
// Initialize machine error classes for FBs/FCs
// The subindex is the functionBlockId in FCLMsgHdlWriteMessageToBuffer
// You can use only four 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;
```

Assigning machine error classes for messages from peripheral devices

If you want to assign the messages from peripheral devices to a machine error class, you have to create it.

Table 7- 4 Initializing machine error classes for messages from peripheral devices

```
//=====================================================================
// 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 Editing user-defined messages

This section contains two examples of the editing of user-defined messages. You must define the user-defined messages in the **fLMsgHdlInit** program unit.

Table 7- 5 Example of a user-defined message no. 4 with four additional values

```
// 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;
```

Table 7- 6 Example of a user-defined message no. 2 with two additional values

```
// 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 Adapting constants in the cPublic library unit

Messages from three FBs/FCs are used in this application example. You must therefore adapt the constants in the **cPublic** library unit.

Table 7- 7 Adapting constants in **cPublic**

```
//=====================================================================
//           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 of 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;
```

After all the adaptations have been made in the units, you have to accept and compile them. You must then save the project and load it to the SIMOTION device.

7.4 Function call

The function calls for user-defined messages and user-defined messages from FBs/FCs are shown in this section.

You define the machine error class on the basis of the event number. You can transfer additional information that is also to be output in the text in the two additional values.

Table 7- 8 Function call for user-defined messages of a function

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

You can define a maximum of four error classes for user-defined messages from FBs/FCs. They are transferred as parameters. You can assign a machine error class to each error class.

Table 7- 9 Function call for user-defined messages in FBs/FCs

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

7.5 Display of the data from the message handling in the symbol browser of SIMOTION SCOUT

Display of messages

After the user has made all the adaptations in the SIMOTION project, the relevant messages and the associated data are displayed in the symbol browser of SIMOTION SCOUT. An example of the most important data is shown in this section. In the example, four system errors and two user-defined messages are present in the project.

In the buffer for active messages, the **pLMsgHdl** program unit, the following variables are displayed in the *gLMsgHdlActiveMessageString* structure:

- Message source in the *asgMessageSource* array

D435.pLMsgHdl:

	Name	Data type	Status value	Display forma
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		

Figure 7-1 Message source

- Message text in the *asgMessageText* array

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]	

Figure 7-2 Message text

- Time stamp in the *asgMessageOccured* array

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]	

Figure 7-3 Time stamp

In the **fLMsgHdl** program unit, the message types are displayed in the *gsLMsgHdlActiveMessageTypes* structure.

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

Figure 7-4 Message types

Overview of the global variables

A

A.1 Variables

Global variables

The following global variables are defined in the message handling:

Table A- 1 Global variables in the message handling

Name	Data type	Unit	Use
gri16LMsgHdlCounterToIniRetainBuffer	INT	pLMsgHdl	A constant is incremented by the script when changes are made in the message handling. This variable is used during startup to decide whether the raw data in the retentive data area (RETAIN) has to be initialized. Retain data is deleted.
gboLMsgHdlInitDriveReady	BOOL	pLMsgHdl	Shows that the initialization software of the message handling has been run through in the BackgroundTask. TRUE: The entire message handling is active FALSE: No messages can be accepted yet
gboLMsgHdlActivateNewMoMaData	BOOL	pLMsgHdl	With TRUE, the information transferred during runtime for modular machines is activated. After activation, the flag is removed by the message handling.
gboLMsgHdlGlobalAcknowledge	BOOL	pLMsgHdl	A global acknowledgement of all active errors in the message handling is triggered with a rising edge. After the acknowledgement, the value is reset to FALSE.
gi32LMsgHdlNumberOfMessageInLog	BOOL	pLMsgHdl	Transfer of the number of the message to be acknowledged (only with single acknowledgement).
gboLMsgHdlStartChangeLanguage	BOOL	pLMsgHdl	Change of the active language for the message handling in STRING format. Start with rising edge. The message handling resets the variable to FALSE after the action.
gu8LMsgHdlActiveLanguage	USINT	pLMsgHdl	Setting of the active language for the message handling. With TRUE, start of the language selection. Is automatically reset by the message handling.

Name	Data type	Unit	Use
gboLMsgHdlStartWriteCompleteMessageLogToStorageMedium			
	BOOL	pLMsgHdl	With a rising edge, the current message log in raw data and STRING format, as well as the information required for the interpretation of the raw data, is written to the storage medium of the SIMOTION device. The message handling resets the variable to FALSE after the action.
gu32LMsgHdlDataSetNoForExportMessageLog			
	UDINT	pLMsgHdl	Name of the file in which the current message log is to be saved Default: *ds000000.dat*
gu8LMsgHdlScrollStep	USINT	pLMsgHdl	The number of messages to be scrolled up or down in the display in SIMOTION IT or in the HMI, can be set here. Default: LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI from **cPublic**
gu8LMsgHdlNumberOfLinesForHMI	USINT	pLMsgHdl	The number of messages to be displayed in SIMOTION IT or HMI can be set here. Default: LMSGHDL_MAX_NUMBER_OF_VISIBLE_LINES_FOR_HMI from **cPublic**
gboLMsgHdlUpdateHMI	BOOL	pLMsgHdl	Update of the active messages display on SIMOTION IT or HMI. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollUp1	BOOL	pLMsgHdl	Scroll up one message in the list of active messages. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollUp	BOOL	pLMsgHdl	Scroll up *gu8ScrollStep* lines in the list of active messages. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollDown1	BOOL	pLMsgHdl	Scroll down one message in the list of active messages.
gboLMsgHdlScrollDown	BOOL	pLMsgHdl	Scroll down *gu8ScrollStep* lines in the list of active messages. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlGoToTop	BOOL	pLMsgHdl	Jump to the start of the active messages.
gboLMsgHdlGoToEnd	BOOL	pLMsgHdl	Jump to the end of the active messages.

Name	Data type	Unit	Use
gsLMsgHdlActiveMsgToHMI	sLMsgHdlHMIActive MsgSgType / sLMsg HdlHMIActiveMsg BaseDataType	pLMsgHdl	List of the active messages that are to be output on SIMOTION IT or HMI.
gboLMsgHdlUpdateHMILog	BOOL	pLMsgHdl	Update of the active messages display on SIMOTION IT or HMI. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollUp1Log	BOOL	pLMsgHdl	Scroll up 1 in the message log list. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollUpLog	BOOL	pLMsgHdl	Scroll up *gu8ScrollStep* lines in the message log list. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollDown1Log	BOOL	pLMsgHdl	Scroll down 1 in the message log list. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlScrollDownLog	BOOL	pLMsgHdl	Scroll down *gu8ScrollStep* lines in the message log list. Performed with rising edge. This is then reset by the message handling.
gboLMsgHdlGoToTopLog	BOOL	pLMsgHdl	Jump to the start of the message log.
gboLMsgHdlGoToEndLog	BOOL	pLMsgHdl	Jump to the end of the message log.
gsLMsgHdlLogMsgToHMI	sLMsgHdlHMIMsg LogSgType/ sLMsg HdlHMIActive MsgBaseDataType	pLMsgHdl	Message log list that is to be output on SIMOTION IT or HMI.
gsLMsgHdlActiveMessagesBaseData	sLMsgHdlActive MessagesBase DataType	pLMsgHdl	Active messages in raw data format.
gi16LMsgHdlNumberOfDOsInProject	INT	pLMsgHdl	Number of drive objects really present in the project.
gsgLMsgHdlMessageLevel	STRING[LMSGHDL_ STRING_LENGTH_OF _MESSAGE_LEVEL]	pLMsgHdl	Auxiliary variable for the decision as to whether the language files have to be loaded again from the storage medium or not, after a restart of the machine.
gsLMsgHdlDefaultMessages	sLMsgHdlDefault MessagesType	pLMsgHdl	List of the system messages currently used in the controller.
gasLMsgHdlToAxisMessages	ARRAY[0..LMSGHDL_ NUMBER_OF_AXES_ ALARM_MESSAGES - 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for axes currently used in the controller.
gasLMsgHdlToFollowingObjects Messages	ARRAY[0..LMSGHDL_ NUMBER_OF_ FOLLOWING_OBJECT _ALARM_MESSAGES- 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for following objects currently used in the controller.

Name	Data type	Unit	Use
gasLMsgHdlToCamsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_CAMS_ALARM_MESSAGES - 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for cams currently used in the controller.
gasLMsgHdlToMeasuringInputsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_MEASURING_INPUTS_ALARM_MESSAGES -1] OF sLMsgHdl TOMessagesType	pLMsgHdl	List of the TO messages for measuring inputs currently used in the controller.
gasLMsgHdlToOutputCamsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_ OUTPUT_CAMS_ ALARM_MESSAGES - 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for output cams currently used in the controller.
gasLMsgHdlToExternalEncodersMessages	ARRAY[0..LMSGHDL_NUMBER_OF_ EXTERNAL_ ENCODERS_ALARM_ MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for external encoders currently used in the controller.
gasLMsgHdlToCamTracksMessages	ARRAY[0..LMSGHDL_NUMBER_OF_CAM_ TRACKS_ALARM_ MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for cam tracks currently used in the controller.
gasLMsgHdlToTemperatureControllersMessages	ARRAY[0..LMSGHDL_NUMBER_OF_ TEMPERATURE_ CONTROLLERS_ ALARM_MESSAGES - 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for temperature controllers currently used in the controller.
gasLMsgHdlToFixedGearsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_FIXED_GEARS_ALARM_ MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for fixed gears currently used in the controller.
gasLMsgHdlToAdditionObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_ ADDITION_OBJECT_ ALARM_MESSAGES - 1] OF sLMsgHdl TOMessagesType	pLMsgHdl	List of the TO messages for addition objects currently used in the controller.

Name	Data type	Unit	Use
gasLMsgHdlToFormulaObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_FORMULA_OBJECT_ALARM_MESSAGES - 1] OF sLMsg HdlTOMessagesType	pLMsgHdl	List of the TO messages for formula objects currently used in the controller.
gasLMsgHdlToSensorsMessages	ARRAY[0..LMSGHDL_NUMBER_OF_SENSORS_ALARM_MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for sensors currently used in the controller.
gasLMsgHdlToControllerObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_CONTROLLER_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for controller objects currently used in the controller.
gasLMsgHdlToPathObjectMessages	ARRAY[0..LMSGHDL_NUMBER_OF_PATH_OBJECT_ALARM_MESSAGES - 1] OF sLMsgHdlTO MessagesType	pLMsgHdl	List of the TO messages for path objects currently used in the controller.
gai16LMsgHdlDOMessageIndex	ARRAY[1..LMSGHDL_MOST_SIGNIFICANT_DO_MESSAGE_NUMBER] OF INT	pLMsgHdl	List of the DO message indices currently used in the controller.
gasgLMsgHdlDOMessages	ARRAY[1..LMSGHDL_NUMBER_OF_DIFFERENT_DO_MESSAGES] OF STRING[LMSGHDL_STRING_LENGTH_OF_MESSAGE_TEXT]	pLMsgHdl	List of the DO messages currently used in the controller.
gsLMsgHdlSystemMessages	sLMsgHdlSystemMessagesType	pLMsgHdl	List of the system messages currently used in the controller.
gasgLMsgHdlAcknowledgeClass	ARRAY OF STRING	pLMsgHdl	List of the messages for the type of acknowledgement currently used in the controller.
gasgLMsgHdlMessageClass	ARRAY OF STRING	pLMsgHdl	List of the messages for the message class currently used in the controller.
gasLMsgHdlMessageFBsFCs	ARRAY OF STRING	pLMsgHdl	List of the messages for message handling messages currently used in the controller.
gasLMsgHdlUserDefinedMessages	ARRAY[0..LMSGHDL_NUMBER_OF_USER_DEFINED_EVENTS - 1] OF sLMsgHdl UserMessagesType	fLMsgHdlInit	List of the user-defined messages currently used in the controller.

Name	Data type	Unit	Use
gsLMsgHdlActiveMessageTypes	sLMsgHdlActive MessageTypesType	fLMsgHdl	This structure displays from which message source messages are active.
gb32LMsgHdlMachineErrorClasses	DWORD	fLMsgHdl	Display of the currently active machine error classes.
gi8LMsgHdlMachineErrorClass	SINT	fLMsgHdl	Display of the machine error class currently with the highest priority.
gab16LMsgHdlEventflag	ARRAY[0..(LMSGHDL_ NUMBER_OF_USER_ DEFINED_EVENTS/16)] OF WORD	fLMsgHdl	Array for display of the active user-defined messages for the message bit handling.
gab16LMsgHdlAckFlag	ARRAY[0..(LMSGHDL_ NUMBER_OF_USER_ DEFINED_EVENTS/16)] OF WORD	fLMsgHdl	Array for display of the user-defined message acknowledgement for the message bit handling.
gasLMsgHdlMessageFBsFCsForHMI	ARRAY[0..LMSGHDL_ NUMBER_OF_ INTERNAL_ APPLICATION_ EVENTS - 1] OF sLMsgHdlMessages FromMessage HandlingType	fLMsgHdl	String texts used for internal message handling messages. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasLMsgHdlDOWithTOInfoForHMI	ARRAY[0..LMSGHDL_ NUMBER_OF_DOS_ WITH_TO - 1] OF sLMsgHdlDOWith TONameType	fLMsgHdl	Names of the DOs with TO. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasLMsgHdlCyclicDOInfoForHMI	ARRAY[0..LMSGHDL_ NUMBER_OF_CYCLIC _DOS - 1] OF sLMsg HdlCyclicDOName-Type	fLMsgHdl	Names of the DOs with cyclic data exchange. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasLMsgHdlAcyclicDOInfoForHMI	ARRAY[0..LMSGHDL_ NUMBER_OF_ ACYCLIC_DOS - 1] OF sLMsgHdlAcyclic DONameType	fLMsgHdl	Names of the DOs without cyclic data exchange. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlAxisNames	ARRAY[0..LMSGHDL_ NUMBER_OF_AXES-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured axes. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlExternalEncoderNames	ARRAY[0..LMSGHDL_ NUMBER_OF_EXTER NAL_ENCODERS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured external encoders. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.

Name	Data type	Unit	Use
gasgLMsgHdlMeasuringInputNames	ARRAY[0..LMSGHDL_NUMBER_OF_MEASURING_INPUTS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured measuring inputs. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlOutputCamNames	ARRAY[0..LMSGHDL_NUMBER_OF_OUTPUT_CAMS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured output cams. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlCamTrackNames	ARRAY[0..LMSGHDL_NUMBER_OF_CAM_TRACKS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured cam tracks. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlCamNames	ARRAY[0..LMSGHDL_NUMBER_OF_CAMS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured cams. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlFollowingObjectNames	ARRAY[0..LMSGHDL_NUMBER_OF_FOLLOWING_OBJECT-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured following objects. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI.
gasgLMsgHdlPathObjectNames	ARRAY[0..LMSGHDL_NUMBER_OF_PATH_OBJECT1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured path objects. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Path.
gasgLMsgHdlFixedGearNames	ARRAY[0..LMSGHDL_NUMBER_OF_FIXED_GEARS-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured fixed gears. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Cam_ext.
gasgLMsgHdlAdditionObjectNames	ARRAY[0..LMSGHDL_NUMBER_OF_ADDITION_OBJECT-1] OF STRING[LMSGHDL_STRING_LENGTH_OF_TO_NAME]	fLMsgHdl	Names of all configured addition objects. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Cam_ext.

Name	Data type	Unit	Use
gasgLMsgHdlFormulaObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ FORMULA_OBJECT-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured formula objects. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Cam_ext.
gasgLMsgHdlSensorNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ SENSORS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured sensors. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Cam_ext.
gasgLMsgHdlControllerObjectNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ CONTROLLER_ OBJECT-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured controller objects. Required in order to be able to interpret relevant messages from buffers in raw data format in the HMI. Only when using the TP Cam_ext.
gasgLMsgHdlTemperatureControl-lerNames	ARRAY[0..LMSGHDL_ NUMBER_OF_ TEMPERATURE_ CONTROLLERS-1] OF STRING[LMSGHDL_ STRING_LENGTH_OF _TO_NAME]	fLMsgHdl	Names of all configured temperature controllers. These are required in order to be able to interpret relevant messages from the buffer in raw data format in the HMI. Only when using TControl.
gsLMsgHdlMessageLogString	sLMsgHdlActive MessageStringType	fLMsgHdl	Message log in STRING format. Only when STRING format has been selected by the script.
grsLMsgHdlMessageLogBaseData	sLMsgHdlMes-sageLogBaseDataType	fLMsgHdl	Message log in raw data format (RETAIN).
grsLMsgHdlMessageLogBaseData-GoneAndOccurred	sLMsgHdlMes-sageLogBaseData-GoneAndOccurred Type	dLMsgHdl	Alternative message variant (not activated by default)

Interpretation of the raw data

B

B.1 Structure

Table B- 1 Table for the interpretation of the raw data in the message log and active messages, part 1

Message source	TO messages	DO messages Error	DO messages Alarm	I/O messages
au8MessageSource [USINT]	1	2	3	4
au8MessageLevel [USINT]	Level [USINT]	Level [USINT]	Level [USINT]	Level [USINT]
au8AcknowledgeClass [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]
au8ErrorClass [USINT]				
au16Parameter1 [UINT]	TO type [USINT]	Axis reference as number [UINT]	Axis reference as number [UINT]	Event class [UINT]
ai16Parameter2 [INT]	TO number [INT]	IO_Id [UINT]	IO_Id [UINT]	Fault ID [UINT]
ab32Parameter3 [DWORD]	Message number [DINT]	Logical address of DO [DINT]	Logical address of DO [DINT]	Logical base address INPUT [DINT]
ab32Parameter4 [DWORD]	AddInfo1_DINT [DINT]	DO number [DINT]	DO number [DINT]	Logical base address OUTPUT [DINT]
ab32Parameter5 [DWORD]	AddInfo2_DINT [DINT]	Error info [DINT]	Alarm info [DINT]	Triggering interrupt [UDINT]
ab32Parameter6 [DWORD]	AddInfo3_DINT [DINT]	Error code [UINT]	Alarm code [UINT]	DP slave diagnostics address [DINT]
ab32Parameter7 [DWORD]	AddInfo4_DINT [DINT]	Type of DO [INT]	Type of DO [INT]	Detailed information [DWORD]
ab32Parameter8 [DWORD]	AddInfo5_DINT [DINT]			Master system ID [UDINT]
ab32Parameter9 [DWORD]				DP slave address [UDINT]
ab32Parameter10 [DWORD]				Slot number [UDINT]

Message source	TO messages	DO messages Error	DO messages Alarm	I/O messages
ab32Parameter11 [DWORD]				Sub-slot number [UDINT]
adtMessageOccured [DT]	Message occurred [DT]	Message occurred [DT]	Message occurred [DT]	Message occurred [DT]
adtMessageGone [DT]	Message gone [DT]	Message gone [DT]	Message gone [DT]	Message gone [DT]

Table B- 2 Table for the interpretation of the raw data in the message log and active messages, part 2

Message source	TimeFault messages	ExecutionFault messages	Message restart	Messages FC/FB
au8MessageSource [USINT]	5	6	7	8
au8MessageLevel [USINT]	Level [USINT]	Level [USINT]	Level [USINT]	Level [USINT]
au8AcknowledgeClass [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]	Type of acknowl-edgement [USINT]
au8errorClass [USINT]				au8ErrorClass [USINT]
au16Parameter1 [UINT]	taskId [UINT]	taskId [UINT]	eventSource (startup) [USINT]	au8toType [USINT]
ai16Parameter2 [INT]				ai16ToNumber [INT]
ab32Parameter3 [DWORD]	Triggering interrupt [UDINT]	Type of processing error [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]
ab32Parameter10 [DWORD]				ai32FunctionBlockId [DINT]

Message source	TimeFault messages	ExecutionFault messages	Message restart	Messages FC/FB
ab32Parameter11 [DWORD]				ab32ErrorCode [DWORD]
adtMessageOccured [DT]	Message occurred [DT]	Message occurred [DT]	Message occurred [DT]	Message occurred [DT]
adtMessageGone [DT]	Message gone [DT]	Message gone [DT]	Message gone [DT]	Message gone [DT]

Table B- 3 Table for the interpretation of the raw data in the message log and active messages, part 3

Message source	User-defined messages	Messages through message handling	DO safety messages
au8MessageSource [USINT]	9	10	11
au8MessageLevel [USINT]	Level [USINT]	Level [USINT]	Level [USINT]
au8AcknowledgeClass [USINT]	Type of acknowledgement [USINT]	Type of acknowledgement [USINT]	Type of acknowledgement [USINT]
au8errorClass [USINT]	u8ErrorClass [USINT]	u8ErrorClass [USINT]	u8ErrorClass [USINT]
au16Parameter1 [UINT]	eventSource [USINT]	eventSource [USINT]	Axis reference as number [INT]
ai16Parameter2 [INT]			IO_Id [UINT]
ab32Parameter3 [DWORD]	eventNumber [INT]	eventNumber [INT]	Logical address of DO [DINT]
ab32Parameter4 [DWORD]	AddInfo1 [DINT]	AddInfo1 [DINT]	DO number [DINT]
ab32Parameter5 [DWORD]	AddInfo2 [DINT]	AddInfo2 [DINT]	Safety info [DINT]
ab32Parameter6 [DWORD]			Safety code [UINT]
ab32Parameter7 [DWORD]			Safety code [UINT]
ab32Parameter8 [DWORD]			
ab32Parameter9 [DWORD]			
ab32Parameter10 [DWORD]		ai32FunctionBlockId [DINT]	
ab32Parameter11 [DWORD]		ab32ErrorCode [DWORD]	

Message source	User-defined messages	Messages through message handling	DO safety messages
adtMessageOccured [DT]	Message occurred [DT]	Message occurred [DT]	Message occurred [DT]
adtMessageGone [DT]	Message gone [DT]	Message gone [DT]	Message gone [DT]

B.2 Common information of all messages

General

The global data buffer of the message information in raw data format is in the *grsLMsgHdlMessageLogBaseData* structure in the **fLMsgHdl** program unit. This information is stored in the retentive data area (RETAIN) and can be evaluated as follows.

The index in which the last entry in the buffer was stored by the message handling is stored in *grsLMsgHdlMessageLogBaseData.i16ActualIndex*. The buffer is a ring buffer that is written by the message handling in ascending order. When the last entry in the message buffer is filled, the next entry is written again at index 0. The buffer is always sorted according to the time stamp **when the message occurred** starting at the latest entry.

Message source

Table B- 4 Contents of the *au8MessageSource[]* cells

Message source	Value [USINT]
Unknown source	0
TO messages	1
DO error	2
DO alarm	3
Peripheral messages	4
TimeFault messages	5
ExecutionFault messages	6
Restart message	7
Messages from FBs/FCs	8
User-defined messages	9
Messages through message handling	10
DO safety message	11

Message level

Table B- 5 Contents of the *au8MessageLevel[]* cells

Message level [STRING]	Value [USINT]
Unknown	0
Fault	1
Error	2
Alarm	3
Information	4
Safety message	5

Acknowledge class

Table B- 6 Contents of the *au8AcknowledgeClass[]* cells

Acknowledge class [STRING]	Value [USINT]
Unknown	0
No acknowledgement	1
Immediately	2
Power On	3
Immediately / Power On	4

Error class

Table B- 7 Contents of the *au8ErrorClass[]* cells

Error class [STRING]	Value [USINT]
Class0	0
Class1	1
Class2	2
Class3	3

Is only written by the **FCLMsgHdlWriteFBFCMessageToBuffer** function.

B.3 Messages of the technology object

TO messages

The following cells are assigned values for messages from TOs

Table B- 8 au16Parameter1 [UINT] = TO type [USINT]

TO type	Value [USINT]
All types of axes	1
Following object	2
Cam	3
Measuring input	4
Output cam	5
External encoder	6
Cam track	7
Temperature controller	8
Fixed gear	9
Addition object	10
Formula object	11
Sensor	12
Controller object	13
Path object	14

Based on the TO type, the TO name in STRING format belonging to the TO number can be read out of the appropriate area from the **fLMsgHdl** program unit. The TO number corresponds to the array index in which the name of the TO is stored.

Table B- 9 ai16Parameter2 [INT] = TO number [INT]

TO type	Value [USINT]	Array with the name of the TO belonging to the number of the TO in the pLMsgHdl program unit
All types of axes	1	gasgLMsgHdlAxisNames
Following object	2	gasgLMsgHdlFollowingObjectNames
Cam	3	gasgLMsgHdlCamNames
Measuring input	4	gasgLMsgHdlMeasuringInputNames
Output cam	5	gasgLMsgHdlOutputCamNames
External encoder	6	gasgLMsgHdlExternalEncoderNames
Cam track	7	gasgLMsgHdlCamTrackNames
Temperature controller	8	gasgLMsgHdlTemperatureControllerNames
Fixed gear	9	gasgLMsgHdlFixedGearNames
Addition object	10	gasgLMsgHdlAdditionObjectNames
Formula object	11	gasgLMsgHdlFormulaObjectNames
Sensor	12	gasgLMsgHdlSensorNames

TO type	Value [USINT]	Array with the name of the TO belonging to the number of the TO in the pLMsgHdl program unit
Controller object	13	gasgLMsgHdlControllerObjectNames
Path object	14	gasgLMsgHdlPathObjectNames

Therefore, for example, the name of the TO type = 2 with TO number = 5 is stored in *gasgLMsgHdlFollowingObjectName[5]* in STRING format.

The following variables contain the numbers of the message belonging to the technology object as well as the possible additional values belonging to the message. This information is required in order to independently combine the appropriate message text including the additional values.

ab32Parameter3 [DWORD] = message number [DINT] (TSI#alarmNumber) number of the technological message

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

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

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

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

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

The maximum five additional values of the technological messages are automatically provided by the system in each of the three data types DINT, UDINT and REAL. So that all additional values do not have to be saved in all data types, the additional values are stored as a bit pattern in a variable of the DWORD data type and have to be converted in the message text depending on the required format. I.e. if a technological message requires that additional value 1 be interpreted in decimal format, additional value 1 from the message log of DWORD must be converted to decimal format before it is integrated in the message text. This applies for all additional values of technological messages in all the possible data formats.

adtMessageOccured[DT] = time when message occurred in DT format

adtMessageGone[DT] = time when message gone in DT format

The other parameters are not assigned.

B.4 Errors on the drive object

Errors on the DO

There are three different types of drive objects (DOs) in the message handling:

- Drive objects with TO axis
- Drive objects with cyclic standard telegram
- Drive objects without cyclic standard telegram

To identify the individual DOs, different information is entered in the message buffer depending on the type.

The required information is stored in the following parameters:

au16Parameter1 [UINT] = axis reference as number [UINT] (TO number of the axis)
ai16Parameter2 [INT] = IO-ID [UINT] (value 0 = input, 1 = output)
ab32Parameter3 [DWORD] = logical address of the DO [DINT]
ab32Parameter4 [DWORD] = DO number [DINT]

These parameters are assigned as follows for the various DO types:

DO with TO axis

au16Parameter1 = number of the axis to which the DO belongs (see TO messages)
ai16Parameter2 = -1 (no IO-ID)
ab32Parameter3 = -1 (no logical address transferred)
ab32Parameter4 = 255 (no DO number transferred)

DO with cyclic standard telegram

au16Parameter1 = 0 (no TO axis assigned)
ai16Parameter2 = 0/1 (logical address is 0 = input or 1 = output)
ab32Parameter3 = logical address of the DO (logical address from HW Config)
ab32Parameter4 = 255 (no DO number transferred)

DO without cyclic standard telegram

au16Parameter1 = 0 (no TO axis assigned)
ai16Parameter2 = 0/1 (logical address is 0 = input or 1 = output)
ab32Parameter3 = a logical address of the device at which the DO is located
ab32Parameter4 = DO number (drive object number from the properties of the DO)

The information specifying the individual DO is stored in the **fLMsgHdl** program unit in the *gasLMsgHdlDOWithTOInfoForHMI*, *gasLMsgHdlCyclicDOInfoForHMI* and *gasLMsgHdlAcyclicDOInfoForHMI* variables.

DO with TO axis

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

Table B- 10 sLMsgHdlDOWithTONameType

Parameter	Data type	Description
sgDOName	STRING[25]	Name of the drive object connected to an axis.
sgCUName	STRING[25]	Name of the control unit on which the DO is located.

DO with cyclic standard telegram

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

Table B- 11 sLMsgHdlCyclicDONameType

Parameter	Data type	Description
i32LogAddress	DINT	Logical address of the drive object.
i16IoId	INT	IO-ID of the logical address. 0 = INPUT 1 = OUTPUT
sgDOName	STRING[25]	Name of the drive object connected to an axis.
sgCUName	STRING[25]	Name of the control unit on which the DO is located.

DO without cyclic standard telegram

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

Table B- 12 sLMsgHdlAcyclicDONameType

Parameter	Data type	Description
i32LogAddress	DINT	A logical address of the device at which the DO is located.
i16IoId	INT	IO-ID of the logical address. 0 = INPUT 1 = OUTPUT
u8DONumber	USINT	DO number of the drive object.
sgDOName	STRING[25]	Name of the drive object connected to an axis.
sgCUName	STRING[25]	Name of the control unit on which the DO is located.

ab32Parameter5 [DWORD] = additional value for the error [DINT] (contents of parameter DOx.r0949)

ab32Parameter6 [DWORD] = error code [UINT] (contents of parameter DOx.r0945)

ab32Parameter7 [DWORD] = type of the DO with error [INT] (contents of parameter DOx.r0107)

adtMessageOccured[DT] = time when message occurred in DT format
adtMessageGone[DT] = time when message gone in DT format

B.5 Warnings on the drive object

DO alarms

The DO information for alarms on drive objects in the message handling is structured in the same way as for DO errors.

ab32Parameter5 [DWORD] = additional value for the alarm [DINT] (contents of parameter DOx.r2124)
ab32Parameter6 [DWORD] = number of the alarm [UINT] (contents of parameter DOx.r2122)
ab32Parameter7 [DWORD] = type of the DO with alarm [INT] (contents of parameter DOx.r0107)
adtMessageOccured[DT] = time when message occurred in DT format
adtMessageGone[DT] = time when message gone in DT format

B.6 Messages on the I/O

Peripheral messages

au16Parameter1 [UINT] = event class [UINT] (TSI#eventClass)
ai16Parameter2 [INT] = fault ID [UINT] (TSI#faultId)
ab32Parameter3 [DWORD] = logical base address of INPUT [DINT] (TSI#logBaseAdrIn)
ab32Parameter4 [DWORD] = logical base address of OUTPUT [DINT] (TSI#logBaseAdrOut)
ab32Parameter5 [DWORD] = triggering interrupt [UDINT] (TSI#interruptId)
ab32Parameter6 [DWORD] = DP slave diagnostics address [DINT] (TSI#logDiagAdr)
ab32Parameter7 [DWORD] = detailed information [DWORD] (TSI#details)
ab32Parameter8 [DWORD] = master system ID of the relevant I/O module [UDINT] (as in HW Config)
ab32Parameter9 [DWORD] = DP slave address [UDINT] (as in HW Config)
ab32Parameter10 [DWORD]= slot number [UDINT] (as in HW Config)
ab32Parameter11 [DWORD]= sub-slot number [UDINT] (as in HW Config)
adtMessageOccured[DT] = time when message occurred in DT format
adtMessageGone[DT] = time when message gone in DT format

B.7 TimeFault messages

TimeFault messages can only occur in the BackgroundTask or a TimerInterruptTask.

Therefore, the information for TimeFault messages is created as follows:

au16Parameter1 [UINT]	= task ID [UINT]
au16Parameter1	= 1 TimeFault in the BackgroundTask
au16Parameter1	= 2 TimeFault in a TimerInterruptTask
ab32Parameter3 [DWORD]	= triggering event [UDINT] (TSI#interruptId)
adtMessageOccured[DT]	= time when message occurred in DT format
adtMessageGone[DT]	= time when message gone in DT format

B.8 ExecutionFault messages

ExecutionFault messages are triggered for program faults. As the SIMOTION device goes into STOP mode after a program fault in a cyclic task, these messages are only taken into the message handling after a restart of the SIMOTION device. These active messages must be acknowledged.

The information for ExecutionFault messages is created as follows:

au16Parameter1 [UINT]	= task ID [UINT] (this value is not supported)
ab32Parameter3 [DWORD]	= type of execution fault [UDINT] (TSI#executionFaultType)
adtMessageOccured[DT]	= time when message occurred in DT format
adtMessageGone[DT]	= time when message gone in DT format

B.9 Messages through startup of the SIMOTION device

The message generated by the message handling at every startup of the SIMOTION device has the following constant assignment:

au8MessageSource	= 7 (message source is restart)
au8MessageLevel	= 4 (note)
au8AcknowledgeClass	= 1 (no acknowledgement required)
adtMessageOccured[DT]	= time when message occurred in DT format (time of restart)
adtMessageGone[DT]	= time when message gone in DT format (time of restart)

The messages for a startup of the SIMOTION device are only entered in the message log. An active message is not generated.

B.10 User-defined messages

These messages are generated by the user within the application by calling the **FCLMsgHdlWriteUserMessageToBuffer** function.

ab32Parameter3 [DWORD]	= number of the user-defined message [DINT]
ab32Parameter4 [DWORD]	= Addinfo1 [DINT]
ab32Parameter5 [DWORD]	= Addinfo2 [DINT]
adtMessageOccured[DT]	= time when message occurred in DT format
adtMessageGone[DT]	= time when message gone in DT format

B.11 User-defined messages for FB/FC and FB units

User-defined messages for FBs/FCs are generated within the application by calling the **FCLMsgHdlWriteFBFCMessageToBuffer** function.

The common information of a message should be interpreted as follows:

au8ErrorClass [USINT]	= error class of the FB/FC, which determines the machine error class, which is set by the FB/FC.
ab32Parameter3 [DWORD]	= number of the user-defined message [DINT]
ab32Parameter4 [DWORD]	= additional value 1 [DINT]
ab32Parameter5 [DWORD]	= additional value 2 [DINT]
ab32Parameter10 [DWORD]	= unique number of the FB/FC is assigned by the user (functionBlockId [DINT])
ab32Parameter11 [DWORD]	= error code of the FB/FC [DWORD]
adtMessageOccured[DT]	= time when message occurred in DT format
adtMessageGone[DT]	= time when message gone in DT format

B.12 Messages through message handling

If an error occurs while processing the message handling, a user-defined message of the message handling is triggered. The messages through the message handling are transferred to the message handling by the **FCLMsgHdlWriteFBFCMessageToBuffer** function. The messages start with event number 100.000 and are numbered consecutively. The text of the message with number 100.000 is then at subindex 0, etc.

The messages through the message handling in STRING format and their structure are stored in the **fLMsgHdl** program unit in the *gasgLMsgHdlMessageFBsFCsForHMI* array.

The array is instantiated as follows:

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

sLMsgHdlMessagesFromMessageHandlingType has the following structure:

Table B- 13 sLMsgHdlMessagesFromMessageHandlingType

Parameter	Data type	Description
sgLMsgHdlTextPart1	STRING[160]	First substring of the message through the message handling
ab8LMsgHdlAdditionalValue1	ARRAY [0..1] OF BYTE	Specification of number and format of the possible first additional value of the message.
sgLMsgHdlTextPart2	STRING[50]	Second substring of the message through the message handling
ab8LMsgHdlAdditionalValue2	ARRAY [0..1] OF BYTE	Specification of number and format of the possible second additional value of the message.
sgLMsgHdlTextPart3	STRING[50]	Third substring of the message through the message handling
ab8LMsgHdlAdditionalValue3	ARRAY [0..1] OF BYTE	Specification of number and format of the possible third additional value of the message.

Contact

C

C.1 Contacts

Siemens AG
Digital Factory
Factory Automation
Production Machines
DF FA PMA APC
Frauenauracher Strasse 80
D-91056 Erlangen, Germany
Fax.: +49 9131 98 1297
tech.team.motioncontrol@siemens.com

C.2 Internet addresses

Additional information on various topics is provided on the following Internet pages.

See also

SIMOTION (www.siemens.com/simotion)

SINAMICS (www.siemens.com/sinamics)

Motion Control / Application Center (www.siemens.com/motioncontrol/apc)

Packaging (www.siemens.com/packaging)

SIMOTION Message Handling
(https://support.industry.siemens.com/cs/ww/en/view/48955585)

SIMATIC S7-1200/S7-1500 and SIMOTION: Acyclic Data Exchange
(https://support.industry.siemens.com/cs/ww/en/view/109479553)

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