

Modicon LMC078

Motion Controller


System Functions and Variables

PLCSystem Library Guide

09/2016

EIO0000001917.01

www.schneider-electric.com

Schneider
 **Electric**

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2016 Schneider Electric. All Rights Reserved.

Table of Contents



	Safety Information	5
	About the Book	7
Chapter 1	LMC078 System Data Types	9
	ST_RetainImageInfo: Additional information on a memory image . . .	9
Chapter 2	LMC078 System Functions	11
2.1	Diagnosis	12
	FC_DiagConfigRead2: Reads the Diagnostic Configuration	13
	FC_DiagConfigSet2: Changes the Diagnostic Configuration	15
	FC_DiagMsgRead: Reads the Diagnostic Information	17
	FC_DiagUserMsgWrite: Triggers the Diagnostic Message	19
	FC_DiagQuit: Resets the Diagnostic Message	20
2.2	IEC_Tasks	21
	FC_CycleCheckSet: Enables/Disables the Diagnostic Messages For the Cycle Time Overrun.	22
	FC_CycleCheckTimeGet: Determines the Time Values for the IEC Task Cycle Time Monitoring	24
	FC_CycleCheckTimeSet: Changes the Time Value for the IEC Task Cycle Time Monitoring.	25
	FC_LzsTaskGetCurrentInterval: Determines the Interval Time of the Task in the IEC Program	26
	FC_LzsTaskGetInterval: Read the Interval Time of the Task in the IEC Program.	28
	FC_PrgResetAndStart: Resets and Starts the Application Using the User Program	29
2.3	LogicalAddress	30
	FC_CompareStLogicalAddress: Compares the Two Logical Addresses, i_stLogAddr1 and i_stLogAddr2.	31
	FC_IsStLogicalAddressValid: Verifies the Validation of the Logical Address	32
2.4	MessageLogger	33
	FC_MsgLogSave: Stores the Message Log File on the Bulk Memory	33
2.5	RemoteFile	35
	FC_RemoteDeviceCreate: Setup Device for File Services	36
	FC_RemoteUserIdSet: Sets User Name and Password for Remote File Services.	38

2.6	Retain	41
	FC_CheckProgramIdent: Verifies If the Memory Image Fits the Application Before Loading It	42
	FC_GetRetainImageInfo: Reads the Additional Information That Has to Be Written in the File of a Memory Image	44
	FC_RetainImageLoad: Loads the Memory Image of the Retain Memory That Is Located in a File in the Retain Memory of the Controller	46
	FC_RetainImageSave: Saves the Contents of the Retain Memory in a File	49
2.7	System	51
	FC_GetNVRamStatus: Verifies Whether the data in the NVRam Was Valid During the Controller Boot Up	52
	FC_GetBootState: Determines If All the Parameters Are Valid After the Restart Process of the Controller	53
	FC_SysReset: Resets the Controller	55
	FC_SysSaveParameter: Saves Custom Parameters to the SD Card	56
	FC_SysShutdown: Helps Ensure that the File System is Secure Before Removing Power from the Controller	57
	FC_UserChangePassword: Changes the User Password	58
2.8	TimeAndSync	60
	FC_GetNsPerCPUClockCycle: Measures Time High-Precisely	61
	FC_GetTimeOfDay: Reads the Current Time of the System in ms Resolution Without the Date	62
	FC_GetTSC: Measures Time High-Precisely	64
2.9	VolumeOperations	65
	FC_GetFreeDiskSpace: Reads Out the Free Memory Space of a Memory Medium	66
	FC_GetTotalDiskSpace: Reads Out the Size of a Memory Medium	69
Glossary		71
Index		73

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in death** or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in death** or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This document will acquaint you with the system functions and variables offered within the LMC078 Motion Controller. The LMC078 PLCSystem library contains functions and variables to get information and send commands to the controller system. This document describes the data types functions and variables of the LMC078 PLCSystem library.

The following basic knowledge is required:

- basic information on functionality, structure and configuration of the LMC078
- programming in the FBD, LD, ST, IL or CFC language
- System Variables (global variables)

Validity Note

This document has been updated for the release of SoMachine V4.2.

Related Documents

Title of Documentation	Reference Number
Modicon LMC078 Motion Controller Programming Guide	<i>EIO0000001909 (ENG)</i> <i>EIO0000001910 (FRE)</i> <i>EIO0000001911 (GER)</i> <i>EIO0000001912 (SPA)</i> <i>EIO0000001913 (ITA)</i> <i>EIO0000001914 (CHS)</i> <i>EIO0000001916 (TUR)</i>

You can download these technical publications and other technical information from our website at <http://download.schneider-electric.com>

Product Related Information

WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.¹
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

¹ For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Chapter 1

LMC078 System Data Types

ST_RetainImageInfo: Additional information on a memory image

Variable Structure

This table describes the parameters of the `ST_RetainImageInfo` system variable (`ST_RetainImageInfo_STRUCT` type):

% MW	Variable name	Type	Comment
n/a	sFileName	STRING[255]	File name of the retain image in the format 8.3.
n/a	dtRetainSaveDateTime	DATE_AND_TIME	Date of creation of the retain image in the Date_and_Time format. If no RTC is available on the controller, then DT#1970-01-01-00:00 is stored.
n/a	sHardwareType	STRING[80]	Controller type.
n/a	sSetRetainSize	STRING[80]	Size of the retain memory.
n/a	sUserInfo	STRING[255]	The text that was specified when creating the retain image.
n/a	sMD5Checksum	STRING[33]	Checksum value of the retain data.
n/a	diLengthRetainData	DINT	Number of the retain data in bytes.

Chapter 2

LMC078 System Functions

Overview

This chapter describes the system functions included in the Modicon LMC078 Motion Controller PLCSystem Library.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Diagnosis	12
2.2	IEC_Tasks	21
2.3	LogicalAddress	30
2.4	MessageLogger	33
2.5	RemoteFile	35
2.6	Retain	41
2.7	System	51
2.8	TimeAndSync	60
2.9	VolumeOperations	65

Section 2.1

Diagnosis

Overview

This section describes the functions under Diagnosis.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_DiagConfigRead2: Reads the Diagnostic Configuration	13
FC_DiagConfigSet2: Changes the Diagnostic Configuration	15
FC_DiagMsgRead: Reads the Diagnostic Information	17
FC_DiagUserMsgWrite: Triggers the Diagnostic Message	19
FC_DiagQuit: Resets the Diagnostic Message	20

FC_DiagConfigRead2: Reads the Diagnostic Configuration

Function Description

The FC_DiagConfigRead2 function reads the diagnostic configuration. The diagnostic class and sub class of the diagnostic number `i_diDiagCode` is provided. The class and sub class are read for either global or individual devices.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
<code>i_diDiagCode</code>	DINT	Diagnostic number.
<code>i_stLogAddr</code>	ST_LogicalAddress	Logical address of the device.

The following table describes the output variables:

Output	Type	Comment
<code>FC_DiagConfigRead2</code>	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Invalid diagnostic number.

The following table describes the input/output variables:

Input/Output	Type	Comment
iq_diDiagClass	DINT	Diagnostic class.
iq_diDiagSubClass	DINT	Diagnostic sub class.

Example

Declaration

```
VAR
    diDiagCode: DINT := 8788;
    diDiagClass: DINT := 0;
    DiagSubClass: DINT := 0;
    xRead: BOOL := FALSE;
END_VAR
```

Program

```
IF xRead THEN
    SEC.FC_DiagConfigRead2;
    i_diDiagCode := diDiagCode;
    i_stLogAddr := MyController.stLogicalAdress;
    iq_diDiagClass := diDiagClass;
    iq_diDiagSubClass := diDiagSubClass;
END_IF
```

FC_DiagConfigSet2: Changes the Diagnostic Configuration

Function Description

Using the FC_DiagConfigSet2 function, you can change the configuration of the diagnostic message with the number `i_diDiagCode`. The configuration of the diagnostic message contains the diagnostic class and the sub class.

The system reaction to the diagnostic message is set through the diagnostic class that is triggered by a certain diagnostic message. The sub class is used for a more precise subdivision of the diagnostic classes.

The diagnostic configuration of a diagnostic message can:

- Be changed in the whole system (for all objects). The input variable `i_stLogAddr` has the value `Gc_stLogAddrAllTypes`.
- Be changed for a certain object or a group of objects. The input variable `i_stLogAddr` contains the logical address of an object or object group. Then the configuration is only performed for a certain object or a group of objects.

If `i_diDiagClass = 3`, `i_diDiagSubClass` has to be specified. Otherwise this input variable is ignored.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
<code>i_diDiagCode</code>	DINT	Number of the diagnostic message that has to be changed.
<code>i_diDiagClass</code>	DINT	New diagnostic class for the diagnostic message.
<code>i_stLogAddr</code>	ST_LogicalAddress	Logical address of the device.
<code>i_diDiagSubClass</code>	DINT	New diagnostic sub class for the diagnostic message.

The following table describes the output variable:

Output	Type	Comment
FC_DiagConfigSet2	DINT	See return value description table below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Invalid diagnostic number.
-2	Invalid diagnostic class.
-4	Invalid diagnostic class (only the class 1, 2 or 3 can be assigned to the diagnostic codes of the drive with the reaction A, B and C).
-461	In the current phase, reading out parameters through the service channel is not supported.
-462	Addressed axis does not support this function.

Example

Declaration

```

VAR
    diDiagCode: DINT := 8788;
    diDiagClass: DINT := 0;
    DiagSubClass: DINT := 0;
    xSet: BOOL := FALSE;
END_VAR

```

Program

```

IF xSet THEN
    SEC.FC_DiagConfigSet2;
    i_diDiagCode := diDiagCode;
    iq_diDiagClass := 2;
    i_stLogAddr := MyController.stLogicalAdress;
    iq_diDiagSubClass := 0;
END_IF

```


FC_DiagMsgRead: Reads the Diagnostic Information

Function Description

The FC_DiagMsgRead function reads and clears the diagnostic buffer (FIFO buffer) of the controller. It waits until a diagnostic message is received or the time specified in `i_diTimeout` has expired. If `i_diTimeout = 0`, the function waits for the message indefinitely.

You can save approximately 320 messages. The oldest message is presented first for reading. If there is no message in the buffer or if the timeout has expired, the function returns zero for all results. Reading the buffer does not delete any pending diagnostic message. You can read all the diagnostic messages with the class 1 to 7 since the start-up (restart) of the controller.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
<code>i_diTimeout</code>	DINT	Maximum waiting period in milliseconds. If <code>i_diTimeout = 0</code> , the function waits for the message indefinitely.

The following table describes the output variable:

Output	Type	Comment
<code>FC_DiagMsgRead</code>	DINT	See the return value description table below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-3	Timeout exceeded.

The following table describes the input/output variables:

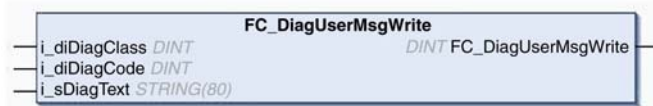
Input/Output	Type	Comment
<code>iq_diNr</code>	DINT	Diagnostic number.
<code>iq_diClass</code>	DINT	Diagnostic class.
<code>iq_stLogAdr</code>	ST_LogicalAddress	Logical address of the controller.

FC_DiagUserMsgWrite: Triggers the Diagnostic Message

Function Description

The FC_DiagUserMsgWrite function allows you to trigger diagnostic messages from the program. The diagnostic class (range 0...4) of the diagnostic message is set via the parameter i_diDiagClass. In DiagCode, the diagnostic number is transferred. For the range 8850...8899 of the system diagnostic messages is reserved. Use i_sDiagText to transfer the diagnostic text. The maximum length is set to 55 characters.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_diDiagClass	DINT	Diagnostic class.
i_diDiagCode	DINT	Diagnostic number.
i_sDiagText	STRING(80)	Diagnostic text (maximum 55 characters).

The following table describes the output variable:

Output	Type	Comment
FC_DiagUserMsgWrite	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-2	Invalid diagnostic class.
-5	Invalid message number.

FC_DiagQuit: Resets the Diagnostic Message

Function Description

The FC_DiagQuit function resets the pending diagnostic message. The diagnostic message is displayed in the parameters `DiagCode` and `DiagMsg`.

Loss of position of the axes and physical encoder (SinCos, incremental encoder) is possible as a result of the error acknowledgment of diagnostic message 8506 (SERCOS Master communication not possible) without restarting the controller.

⚠ CAUTION

POSITION LOSS DUE TO BUS ERROR

Acknowledge the diagnostic message only after reinitialization or homing of the system.

Failure to follow these instructions can result in injury or equipment damage.

NOTE: You can reinitialize the system by a power cycle or by the program using the functions `FC_SysReset()` or `FC_PrgResetAndStart()`.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_DiagQuit	DINT	See the return value description table below.

The following table describes the return value:

Value	Description
0	Action is successfully completed.

Section 2.2

IEC_Tasks

Overview

This section describes the functions under IEC_Tasks.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_CycleCheckSet: Enables/Disables the Diagnostic Messages For the Cycle Time Overrun	22
FC_CycleCheckTimeGet: Determines the Time Values for the IEC Task Cycle Time Monitoring	24
FC_CycleCheckTimeSet: Changes the Time Value for the IEC Task Cycle Time Monitoring	25
FC_LzsTaskGetCurrentInterval: Determines the Interval Time of the Task in the IEC Program	26
FC_LzsTaskGetInterval: Read the Interval Time of the Task in the IEC Program	28
FC_PrgResetAndStart: Resets and Starts the Application Using the User Program	29

FC_CycleCheckSet: Enables/Disables the Diagnostic Messages For the Cycle Time Overrun

Function Description

The `FC_CycleCheckSet` function enables/disables the diagnostic message 8317 cycle time overrun.

If `i_xCheck = FALSE`, the diagnostic messages are disabled.

If `i_xCheck = TRUE`, the diagnostic messages are enabled.

The diagnostic messages get activated when the next cycle starts.

NOTE: When this function is called, an entry is made in the message logger.

NOTE: You should use the function `FC_CycleCheckTimeSet()` instead to keep the watchdog functional.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
<code>i_xCheck</code>	BOOL	FALSE: Disables the diagnostic messages. TRUE: Enable the diagnostic messages.

The following table describes the output variable:

Output	Type	Comment
FC_CycleCheckSet	DINT	See the return value description table below.

The following table describes the return value:

Value	Description
0	Action is successfully completed.

FC_CycleCheckTimeGet: Determines the Time Values for the IEC Task Cycle Time Monitoring

Function Description

The FC_CycleCheckTimeGet function provides the time values for the IEC task cycle time check functions of the IEC task calling it.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variables:

Input/Output	Type	Comment
FC_CycleCheckTimeGet	DINT	See the return value description table below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Incorrect parameter.

The following table describes the input/output variables:

Input/Output	Type	Comment
iq_diOverrunTime	DINT	Value for the standard cycle time monitoring in milliseconds.
iq_diWDogFactor	DINT	Value for the watchdog factor.

FC_CycleCheckTimeSet: Changes the Time Value for the IEC Task Cycle Time Monitoring

Function Description

The `FC_CycleCheckTimeSet` function manipulates the time values for the IEC task cycle time check functions of the IEC task calling it.

The task continues to be called at the interval defined in the task configuration.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variables:

Input	Type	Comment
<code>i_diOverrunTime</code>	DINT	Value for the standard cycle time monitoring in ms. It determines the value for default cycle time check, in ms (Diagnosis message 8317 cycle time overrun).
<code>i_diWDogFactor</code>	DINT	Value for the watchdog factor. It determines the value for a serious cycle time error detected (diagnostic message 313 excessive cycle time overrun).

The following table describes the output variable:

Output	Type	Comment
<code>FC_CycleCheckTimeSet</code>	DINT	See the return value description table below.

The following table describes the return value:

Value	Description
0	Action is successfully completed.

FC_LzsTaskGetCurrentInterval: Determines the Interval Time of the Task in the IEC Program

Function Description

The FC_LzsTaskGetCurrentInterval function provides a way to read the configured interval of an external task. Depending on the trigger the interval can be a constant or variable. The currently configured interval is given in ms. The function returns -1 for tasks with a variable interval.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_LzsTaskGetCurrentInterval	DINT	See the return value description table below.

The following table describes the return values:

Trigger	Value
INIRQ1	-1
INIRQ2	-1
INIRQ3	-1
INIRQ4	-1
INIRQ1_4	-1
RTP_READ	Provides the Sercos cycle time in Sercos phase 4.
RTP_MENC	Provides the Sercos cycle time in Sercos phase 4.
RTP_LENC	Provides the Sercos cycle time in Sercos phase 4.
RTP_AXIS	Provides the Sercos cycle time in Sercos phase 4.
MDT_WRITE_ACCESS	Provides the RTP value in Sercos phase 4.
NOTE: -1 means that the task has a variable interval.	

Example

A task with 50 ms cycle time was projected. The system is to read the current interval of the task.

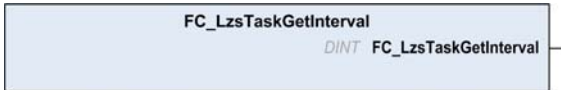
```
diIntervalMs:DINT:=0;
diCurrentIntervalUs:DINT:=0;
diIntervalMs := FC_LzsTaskGetInterval();
(* provides 50 * )
diCurrentIntervalUs := FC_LzsTaskGetCurrentInterval() ;
(* provides 50000 * )
```

FC_LzsTaskGetInterval: Read the Interval Time of the Task in the IEC Program

Function Description

The `FC_LzsTaskGetInterval` function returns the configured interval time of the calling task in ms.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
<code>FC_LzsTaskGetInterval</code>	DINT	See the return value description table below.

The following table describes the return value:

Value	Description
<code>>=0</code>	Interval time in ms.

FC_PrgResetAndStart: Resets and Starts the Application Using the User Program

Function Description

The FC_PrgResetAndStart function starts an asynchronous task that performs the reset (warmstart) and start of the application. All tasks finish their current cycle, that is, even the user code located downstream of the function FC_PrgResetAndStart() is processed. The asynchronous task runs with maximum IEC task priority.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_PrgResetAndStart	DINT	See the return value description table below.

The following table describes the return value:

Value	Description
0	Action is successfully completed.

Section 2.3

LogicalAddress

Overview

This section describes the functions under LogicalAddress.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_CompareStLogicalAddress: Compares the Two Logical Addresses, <code>i_stLogAddr1</code> and <code>i_stLogAddr2</code> .	31
FC_IsStLogicalAddressValid: Verifies the Validation of the Logical Address	32

FC_CompareStLogicalAddress: Compares the Two Logical Addresses, i_stLogAddr1 and i_stLogAddr2.

Function Description

The FC_CompareStLogicalAddress function compares the logical address i_stLogAddr1 with the logical address i_stLogAddr2. If both are identical, then TRUE is returned, otherwise FALSE.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_stLogAddr1	ST_LogicalAddress	Logical address 1
i_stLogAddr2	ST_LogicalAddress	Logical address 2

The following table describes the output variable:

Output	Type	Comment
FC_CompareStLogicalAddress	BOOL	See the return value description table below.

The following table describes the return value:

Value	Description
TRUE	Address 1 is identical to address 2.
FALSE	Address 1 is not identical to address 2.

FC_IsStLogicalAddressValid: Verifies the Validation of the Logical Address

Function Description

The FC_IsStLogicalAddressValid function verifies the validation of the logical address in i_stLogAddr.

TRUE = valid and FALSE = invalid.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_stLogAddr	ST_LogicalAddress	Logical address that has to be verified on validation.

The following table describes the output variable:

Output	Type	Comment
FC_IsStLogicalAddressValid	BOOL	See the return value table description below.

The following table describes the return values:

Value	Description
TRUE	Address is valid.
FALSE	Address is invalid.

Section 2.4

MessageLogger

FC_MsgLogSave: Stores the Message Log File on the Bulk Memory

Function Description

The `FC_MsgLogSave` function stores the content of the message log files as a file on the bulk memory (for example, SD Card) under the name `i_sFilename`. The file name extension, `.mel` is assigned by default. A default name consists of one to eight characters (A... Z, 0... 9). The system appends the device name `ide0:` and the file name extensions.

A complex name consists of device name, file name, and file name extension. If `i_xReset` is set to `TRUE`, the content of the message log file is deleted after having stored the file. Messages that occur while the `FC_MsgLogSave` function is being processed are not lost.

NOTE: The processing time of this function is a few hundred ms. When you use this function you must increase the watchdog time with the `FC_CycleCheckTimeSet`.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input/Output	Type	Comment
i_sFilename	STRING [80]	Name of the saved file.
i_xReset	BOOL	If TRUE, the contents of the message log file is deleted after saving the file (as i_sFilename).

The following table describes the output variable:

Output	Type	Comment
FC_MsgLogSave	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	File is not written.

Section 2.5

RemoteFile

Overview

This section describes the functions under RemoteFile.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_RemoteDeviceCreate: Setup Device for File Services	36
FC_RemoteUserIdSet: Sets User Name and Password for Remote File Services	38

FC_RemoteDeviceCreate: Setup Device for File Services

Functional Description

The `FC_RemoteDeviceCreate` function creates a remote file device with the name `i_sDevName`. You can create a network device for every remote computer whose data has to be reached. The remote computer is specified with the IP address `i_sHostIpAddr`.

A network device name follows the computer device name with a colon. The files are accessed using FTP client services. Configure an FTP server to access a remote computer. When you open a remote file, the entire file is copied using the network to a local buffer. An empty local buffer opens if a remote file is created. The read, write, or `iotcl` access operations are performed on the local copy of the file. When the local copy of a file which is open for editing is closed, it is sent back to the remote computer using the network.

NOTE: When you access the RemoteDevice, the subdirectories may not contain the root string.

NOTE: The processing time of this function is a few hundred ms. When you use this function you must increase the watchdog time with the `FC_CycleCheckTimeSet`.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
<code>i_sDevName</code>	STRING [80]	Name of the remote file device.
<code>i_sHostIpAddr</code>	STRING [80]	IP address of the remote computer.

The following table describes the output variable:

Output	Type	Comment
<code>FC_RemoteDeviceCreate</code>	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Error detected.

Example

```
FC_RemoteUserIdSet('otto','secret');  
FC_RemoteDeviceCreate('usr:', '190.201.100.99');  
pHandle := SysFileOpen('usr:/myfile',AM_WRITE,pResult);
```

FC_RemoteUserIdSet: Sets User Name and Password for Remote File Services

Functional Description

The FC_RemoteUserIdSet function sets the user name and password with which you can access the remote computer. Name (i_sName) and password (i_sPassword) are used to acquire remote access through the FTP services. The user name and password of the controller default values are USER and USER respectively.

NOTE: For important information on Internet security, refer to the *LMC078 Motion Controller Programming Guide (see Modicon LMC078, Motion Controller, Programming Guide)*.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sName	STRING [80]	User name.
i_sPassword	STRING [80]	Password.

The following table describes the output variable:

Output	Type	Comment
FC_RemoteUserIdSet	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Error detected.

Example**Declaration**

```

PROGRAM RemoteFile
VAR
    bWriteTest: BOOL := FALSE;
    bReadTest: BOOL := FALSE;
    hReadFileId: RTS_IEC_HANDLE := 0;
    hWriteFileId: RTS_IEC_HANDLE := 0;
    lResultWriteToFile: UDINT := 99;
    lResultFC_ReadFromFile: UDINT := 99;
    lResultFC_CloseFile: UDINT := 99;
    lResultFC_RemoteDeviceCreate: DINT := 99;
    lResultFC_RemoteUserIdSet: DINT := 99;
    pszRemoteDeviceName: STRING := 'remote0: ';
    pszRemoteIpAddress: STRING := '10.128.156.14';
    pszRemoteFtpUserId: STRING := 'USER';
    pszRemoteFtpPassword: STRING := 'PASSWORD';
    pszFileName: STRING := 'remote0:/myfile.txt';
    pszOutText: STRING := 'Schneider Electric LMC078';
    pszInText: STRING;
    lTextSize: UDINT := 25
    bInitOk: BOOL := TRUE;
    pResult: RTS_IEC_RESULT := 0;
END_VAR

```

Program

```

IF NOT bInitOk THEN
    FC_CycleCheckTimeSet(1000, 60);
    lResultFC_RemoteUserIdSet := FC_RemoteUserIdSet(pszRemoteFtpUserId, pszRemoteFtpPassword);
    lResultFC_RemoteDeviceCreate := FC_RemoteDeviceCreate(pszRemoteDeviceName, pszRemoteIpAddress);
    bInitOk := TRUE;
END_IF

```

```
IF bWriteTest THEN
    hWriteFileId := SysFile.SysFileOpen(pszFileName,AM_WRITE,pResult);
    IF hWriteFileId > 0 THEN
        lResultFC_WriteToFile := SysFile.SysFileWrite(hWriteFileId,
ADR(pszOutText), lTextSize, pResult);
        lResultFC_CloseFile := SysFile.SysFileClose(hWriteFileId);
    END_IF
    bWriteTest:= FALSE;
END_IF
IF bReadTest = TRUE THEN
    hReadFileId := SysFile.SysFileOpen(pszFileName,AM_READ,pResult);
    IF hReadFileId > 0 THEN
        lResultFC_ReadFromFile := SysFile.SysFileRead(hReadFileId,ADR(pszI
nText), lTextSize, pResult);
    lResultFC_CloseFile := SysFile.SysFileClose(hWriteFileId);;
    END_IF
    bReadTest := FALSE;
END_IF
```

Section 2.6

Retain

Overview

This section describes the functions that operate on the retain memory of the controller.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_CheckProgramIdent: Verifies If the Memory Image Fits the Application Before Loading It	42
FC_GetRetainImageInfo: Reads the Additional Information That Has to Be Written in the File of a Memory Image	44
FC_RetainImageLoad: Loads the Memory Image of the Retain Memory That Is Located in a File in the Retain Memory of the Controller	46
FC_RetainImageSave: Saves the Contents of the Retain Memory in a File	49

FC_CheckProgramIdent: Verifies If the Memory Image Fits the Application Before Loading It

Functional Description

The `FC_CheckProgramIdent` function reads the `ProgramIdent` of the memory image from the specified file and compares it with the `ProgramIdent` of the application currently located in memory. Use this function before you use the `FC_RetainImageLoad` function.

NOTE: The typical runtime of the function on a controller is 50 ms if the retain data file is located on the SD card.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input/Output	Type	Comment
<code>i_sRetainImageFileName</code>	STRING [80]	Name of the file whose <code>ProgramIdent</code> is to be compared with the <code>ProgramIdent</code> of the IEC program currently located in the main memory. The file name automatically receives the extension <code>.ret</code> . The file may be located in any existing directory of the controller. The file name cannot include any wildcards.

The following table describes the output variable:

Output	Type	Comment
<code>FC_CheckProgramIdent</code>	DINT	See the value description table given below.

The following table describes the return values:

Value	Description
0	The ProgramIdent of IEC program and memory image match.
-1	The ProgramIdent could not be compared as an invalid file name was transferred. The file name is either too long or it contains one of the characters - :, \, /, * or ?.
-2	The additional information could not be read as the directory specified does not exist.
-3	The ProgramIdent could not be compared, as the file name specified does not exist.
-4	The ProgramIdent of IEC program and memory image do not match.
-5	The ProgramIdent could not be compared as the file specified is not a memory image file of a retain memory.
-6	The ProgramIdent could not be compared as the file specified does not contain a memory image of the retain memory.

Examples

```
diResult := FC_CheckProgramIdent('carton');
```

The ProgramIdent of the application is compared with the ProgramIdent stored in the file 'carton.ret'. The file 'carton.ret' is located on the SD card.

```
diResult := FC_CheckProgramIdent('ide0:\retain\plister');
```

The ProgramIdent of the application is compared with the ProgramIdent stored in the file 'plister.ret'. The file 'plister.ret' is located in the directory 'retain' on the SD card.

```
diResult := FC_CheckProgramIdent('remote:ide0:\label');
```

The ProgramIdent of the application is compared with the ProgramIdent stored in the file 'label.ret'. The file 'label.ret' is located on the on the remote device 'remote' on the SD card.

FC_GetRetainImageInfo: Reads the Additional Information That Has to Be Written in the File of a Memory Image

Function Description

The FC_GetRetainImageInfo function reads the additional information on a memory image from the specified file and stores it in the structure transferred. The additional information is:

- The name of the file
- The memory date
- The controller type
- The size of the retain memory of the controller
- The user information
- The Checksum value
- The size of the memory image

NOTE: If the retain data file is located on drive ide0:, the typical runtime of the function in a controller is 50 ms.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sRetainImage- FileName	STRING [80]	Name of the file whose ProgramIdent is to be compared with the ProgramIdent of the IEC program currently located in the main memory. The file name automatically receives the extension.ret. The file may be located in any existing directory of the controller. The file name must not include any wild cards.

The following table describes the output variable:

Output	Type	Comment
FC_GetRetainImage- Info	DINT	See the value description table given below.

The following table describes the return values:

Value	Description
0	The additional information could be read successfully from the file specified.
-1	The ProgramIdent could not be compared as an invalid file name was transferred. The file name is either too long or it contains one of the characters - :, \, /, * or ?.
-2	The additional information could not be read as the directory specified does not exist.
-3	The additional information could not be read as the file name specified does not exist.
-4	The additional information could not be read as the file specified is not a memory image file of a retain memory.
-5	The additional information could not be read as a general error has been detected in the function FC_GetRetainImageInfo().

The following table describes the input/output variable:

Input/Output	Type	Comment
iq_stInfo	ST_RetainImageInfo	Transfer of additional information.

FC_RetainImageLoad: Loads the Memory Image of the Retain Memory That Is Located in a File in the Retain Memory of the Controller

Functional Description

The FC_RetainImageLoad function reads the memory image from the file specified and stores the same in the retain memory. You can select whether to verify the ProgramIdent or not.

The ProgramIdent is a unique identifier for an application. It is required that the retain memory and the configuration of the retain variables of the application match each other. This is confirmed by a verification of the ProgramIdent. Therefore, the ProgramIdent of the memory image and of the application currently located in the memory are compared and if there is a match, the memory image is loaded.

A new ProgramIdent of an application created upon each change. You can therefore reject a verification of the ProgramIdent since only minor changes have been effected in the code and the declaration of the retain variables has not changed.

The retain memory and the application have to match each other for proper functioning of the controller. Verify the size of the memory image and the size of the retain memory required by the application. The memory image is loaded only if both size values match.

NOTE: The typical runtime of the function in a controller is 100 ms if the retain data file is located on SD card.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION
Do not access the retain data during the execution of the function FC_RetainImageLoad().
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input/Output	Type	Comment
i_sRetainImage- FileName	STRING [80]	Name of the file from which the memory image of the retain memory is to be loaded. The file name automatically receives the extension.ret. The file may be stored into any existing directory of the controller. The file name must not include any wildcards.
i_xProgramIdent- Check	BOOL	You can select whether to verify the ProgramIdent or not. TRUE: The memory image is loaded only if the ProgramIdent of the memory image and the running application match. FALSE: The ProgramIdent is not verified. If you do not verify whether the ProgramIdent matches the application, make sure the application and the memory image of the retain memory to be loaded, match each other. Otherwise, a controller error is detected.

The following table describes the output variable:

Output	Type	Comment
FC_RetainImageLoad	DINT	See the value description table given below.

The following table describes the return values:

Value	Description
0	The memory image has been successfully loaded into retain memory.
-1	The memory image could not be loaded. An invalid file name was specified. The file name is either too long or it contains one of the characters - :, \, /, * or ?.
-2	The memory image could not be loaded as the directory specified does not exist.
-3	The memory image could not be loaded as the file name specified does not exist.
-4	The memory image could not be loaded into retain memory as the ProgramIdent of the memory image does not match the ProgramIdent of the application currently located in memory.
-5	The memory image could not be loaded into retain memory as the data in the memory image are not valid. The data was probably manually changed. This was noticed during a verification of the size of the memory image with the size of the retain data stored in the file.
-6	The memory image could not be loaded into retain memory as the files in the memory image are not valid. This was found during verification of the Checksum value.

Value	Description
-7	The memory image could not be loaded into retain memory as the file specified is not a memory image file of a retain memory.
-8	The memory image could not be loaded into retain memory as a general error has been detected in the function <code>FC_RetainImageLoad()</code> .

Examples

```
diResult := FC_RetainImageLoad('carton', TRUE) ;
```

The retain memory is loaded from the file 'carton.ret' on the SD card. The ProgramIdent is verified.

```
diResult := FC_RetainImageLoad('ide0:\retain\plister', FALSE) ;
```

The retain memory is loaded from the file 'plister.ret' located in the directory 'retain' on the SD card. The ProgramIdent is not verified.

```
diResult := FC_RetainImageLoad('remote:ide0:\label', TRUE) ;
```

The retain memory is loaded from the file 'label.ret', which is located on the SD card plugged on the 'remote' motion controller. The ProgramIdent is verified.

FC_RetainImageSave: Saves the Contents of the Retain Memory in a File

Functional Description

The FC_RetainImageSave function stores the entire content of the retain memory into the file specified. As the addressing and assignment of the variables in the retain memory cannot be predicted, the entire retain memory is stored. Additionally, own information can be stored with the file. If a file with the same name and storage location exists, this file is overwritten.

NOTE: Do not change the retain data while the function is executed. This can result in inconsistent retain data if such data are loaded later. Make sure that the retain data are not accessed during the execution of the function FC_RetainImageSave().

NOTE: The typical runtime of the function in a controller is 400 ms for storing the retain memory on SD card.

NOTE: The processing time of this function is a few hundred ms. When you use this function you must increase the watchdog time by using the FC_CycleCheckTimeSet. For example, CycleCheckTimeSet(500, 1000).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sRetainImage- FileName	STRING[80]	Name of the file in which the content of the retain memory is supposed to be stored. The file name automatically receives the extension.ret. The file may be stored into any existing directory of the controller. The file name must not include any wildcards.
i_sUserInfo	STRING[255]	Information that also has to be stored in the file of the memory image of the retain memory.

The following table describes the output variable:

Output	Type	Comment
FC_RetainImageSave	DINT	See the value description table given below.

The following table describes the return values:

Value	Description
0	The retain memory has been successfully stored.
-1	The retain memory could not be stored as an invalid file name was specified. The file name is either too long or it contains one of the characters - :, \, /, * or ?.
-2	The retain memory could not be stored as the directory does not exist in which the memory image was to be stored.
-3	The retain memory could not be stored, as a general error has been detected in the function FC_RetainImageSave().

Examples

```
diResult := FC_RetainImageSave('carton', 'Dump Cartoner');
```

The retain memory is saved to the file, 'carton.ret' on the SD card.

```
diResult := FC_RetainImageSave('ide0:\retain\plister' ,Speicherabild Plister');
```

The retain memory is saved to the file 'plister.ret' in the directory, 'retain' on the SD card.

```
diResult := FC_RetainImageSave('remote:ide0:\label' ,Dump Labeler');
```

The retain memory is saved to the file, 'label.ret', which is located on the SD card plugged on the 'remote' motion controller.

Section 2.7

System

Overview

This section describes the functions under System.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_GetNVRamStatus: Verifies Whether the data in the NVRam Was Valid During the Controller Boot Up	52
FC_GetBootState: Determines If All the Parameters Are Valid After the Restart Process of the Controller	53
FC_SysReset: Resets the Controller	55
FC_SysSaveParameter: Saves Custom Parameters to the SD Card	56
FC_SysShutdown: Helps Ensure that the File System is Secure Before Removing Power from the Controller	57
FC_UserChangePassword: Changes the User Password	58

FC_GetNVRamStatus: Verifies Whether the data in the NVRam Was Valid During the Controller Boot Up

Functional Description

The FC_GetNVRamStatus function reads the status of NVRam while booting. This is important if the power is removed from the controller for an extended period. If the NVRam verification at boot is unsuccessful, by default the system makes an entry into the message logger (316 NvRam CRC error detected).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_GetNVRamStatus	BOOL	See the return value description table given below.

The following table describes the return values:

Value	Description
TRUE	NVRam verification at boot was successful.
FALSE	Error detected when verifying the NVRam.

FC_GetBootState: Determines If All the Parameters Are Valid After the Restart Process of the Controller

Functional Description

The FC_GetBootState function returns the restart status of the controller. This is necessary as the IEC program is already loaded before the end of the restart process and started, depending on the parameter AutoRun.

The FC_GetBootState() function allows you to verify whether the restart process is complete. Until the completion of restart process, the initialization of the system is not considered as complete and all objects (for example, logical encoders) and all parameters are not valid.

A good use of this function is to make sure that the physical encoder is initialized and functions properly. For example, the duration of the restart process depends on the size of the project.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_GetBootState	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Restart process is not yet complete.
1	Restart process is complete.

Example

```
CASE lState OF
1:
    lBootReady:=FC_GetBootState();
    IF lBootReady=1 THEN
        lState:=lState+1;
    END_IF;
2:
    ....;
END_CASE;
```

FC_SysReset: Resets the Controller

Functional Description

The `FC_SysReset` function restarts the controller by doing a reboot command. For more information, refer to Commanding State Transitions (see *Modicon LMC078, Motion Controller, Programming Guide*).

NOTE: Make sure that the controller boot process is complete before requesting a restart with the `FC_SysReset()` function. You can implement this using the `FC_GetBootState()` function.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
<code>FC_SysReset</code>	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Controller was not reset.

FC_SysSaveParameter: Saves Custom Parameters to the SD Card

Functional Description

The `FC_SysSaveParameter` function saves the configurable parameters to the SD card. This file is used to initialize the system on boot.

NOTE: The processing time of this function is a few hundred ms. When you use this function you must increase the watchdog time by using the `FC_CycleCheckTimeSet`. For example, `CycleCheckTimeSet(500, 1000)`.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
<code>FC_SysSaveParameter</code>	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Error detected.

FC_SysShutdown: Helps Ensure that the File System is Secure Before Removing Power from the Controller

Functional Description

The FC_SysShutdown function helps ensure that the controller closes the file system and all the open files.

It first triggers a diagnostic message, 8023 (Controller shutdown) for the message logger and stops all the IEC tasks. It then closes the file system and open files, after which it stops the CPU. You can now disconnect the system from the power supply.

A hardware reset on the front panel of the controller or a power OFF/power ON of the control voltage is required to restart the system again.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_SysShutdown	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Controller has not been shut down correctly.

FC_UserChangePassword: Changes the User Password

Functional Description

The FC_UserChangePassword function changes the password for an existing user. The username can be used for remote file services. You must remember the existing password.

NOTE: For important information on password security, refer to the *LMC078 Motion Controller Programming Guide (see Modicon LMC078, Motion Controller, Programming Guide)*.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sUserName	STRING(80)	User name.
i_sCurrentPassword	STRING(80)	Current password.
i_sNewPassword	STRING(80)	New password.

The following table describes the output variable:

Output	Type	Comment
FC_UserChangePassword	DINT	See the return value description table given below.

The following table describes the return values:

Value	Description
0	Action is successfully completed.
-1	Internal error.
-2	Unknown user.
-3	Invalid password.

Section 2.8

TimeAndSync

Overview

This section describes the functions under TimeAndSync.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_GetNsPerCPUClockCycle: Measures Time High-Precisely	61
FC_GetTimeOfDay: Reads the Current Time of the System in ms Resolution Without the Date	62
FC_GetTSC: Measures Time High-Precisely	64

FC_GetNsPerCPUClockCycle: Measures Time High-Precisely

Function Description

The `FC_GetNsPerCPUClockCycle` function outputs the length of time for a CPU timer cycle in ns. You can use this value to convert into ns the value that the `FC_GetTSC` function returns.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
<code>FC_GetNsPerCPUClockCycle</code>	LREAL	Number of ns per CPU clock cycle.

Example

Declaration:

```
ulStartClock:UDINT;
ulEndClock:UDINT;
ulClock:UDINT;
lrTimeUs:LREAL;
```

Program:

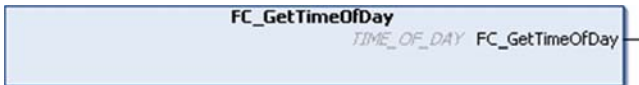
```
ulStartClock := FC_GetTSC();
:<Here put the code you want to measure the time execution>
ulEndClock := FC_GetTSC();
ulClock := ulEndClock - ulStartClock;
lrTimeUs:= UDINT_TO_LREAL(ulClock) * FC_GetNsPerCPUClockCycle() /1000.0
;
```

FC_GetTimeOfDay: Reads the Current Time of the System in ms Resolution Without the Date

Function Description

The FC_GetTimeOfDay function reads the current time of the system in ms resolution without the date. A time stamp with date specification in ms is required in the IEC program. It returns the time in the TimeOfDay format.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_GetTimeOfDay	TIME_OF_DAY	Time in TimeOfDay format (range from 00 h00m00s000ms to 23 h59min59s999ms)

Example

To integrate a logging function in the IEC program, the log is to record diagnostic messages of the system and customer-specific messages. Every recorded message is to be given a time stamp in ms similar to that in the message logger.

The time stamp has to be in the YYYY-MM-DD-hh:mm:ss.000 format. The subsequent function saves the date and time in two variables. These two variables are converted to strings and combined.

```

VAR
    dtActualDate : DATE;
    todActualTimeOfDay: TOD;
    sActualDate : STRING;
    sActualTimeOfDay : STRING;
    sTimestamp : STRING;
END_VAR
    
```

```
dtActualDate:= DT_TO_DATE(MyController.Realtimeclock) ;
todActualTimeOfDay:= FC_GetTimeOfDay() ;
sActualDate := DATE_TO_STRING(dtActualDate) ;
sActualDate := DELETE(sActualDate,2,1);
sActualTimeOfDay := TOD_TO_STRING(todActualTimeOfDay);
sActualTimeOfDay := DELETE(sActualTimeOfDay,4,1) ;
sTimestamp := CONCAT(sActualDate,'-') ;
sTimestamp := CONCAT(sTimestamp,sActualTimeOfDay) ;
```

This function presents how to get the current time in millisecond resolution and the current date from the parameter `RealTimeClock` and the method `FC_GetTimeOfDay()`. You can now save or display the time as a string with a message.

FC_GetTSC: Measures Time High-Precisely

Function Description

The FC_GetTSC function reads the lower-value 32 bit from the internal CPU timer. The Pentium processor has a 64-bit timer that is increased by every clock cycle.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the output variable:

Output	Type	Comment
FC_GetTSC	UDINT	Current counter status of the CPU timer (lower-value 32 bit).

Example

See function FC_GetNsPerCPUClockCycle (*see page 61*).

Section 2.9

VolumeOperations

Overview

This section describes the functions under VolumeOperations.

What Is in This Section?

This section contains the following topics:

Topic	Page
FC_GetFreeDiskSpace: Reads Out the Free Memory Space of a Memory Medium	66
FC_GetTotalDiskSpace: Reads Out the Size of a Memory Medium	69

FC_GetFreeDiskSpace: Reads Out the Free Memory Space of a Memory Medium

Function Description

The FC_GetFreeDiskSpace function reads out the free memory space of a memory medium (flash disk, RAM disk, USB device) in bytes. The name of the memory medium is transferred:

- SD card = ide0:
- RAM disk = ram0:
- USB device = usb2msd:0 (without partition table) or usb2msd:1 (with partition table).

The free memory space of a remote device cannot be read out. If a remote device is specified as parameter, the function returns -1.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sVolumeName	STRING [80]	Name of the device whose free memory space shall be read out

The following table describes the output variable:

Output	Type	Comment
FC_GetFreeDiskSpace	DINT	See the return value description table below.

The following table describes the return values:

Value	Description
0	Free memory space was read out successfully.
-1	Error detected when reading the free memory, for example, an invalid device or remote device was selected

Value	Description
-318	Invalid parameter (i_sVolumeName).

The following table describes the input/output variables:

Input/Output	Type	Comment
iq_uliFreeDiskSpace	ULINT	Free memory space in byte.

Example

Requirement:

The free memory space on a USB device has to be determined. For this, the USB device has to be detected successfully.

Declaration

```
PROGRAM FreeUSBDiskSpace
```

```
VAR
```

```
    uliFreeDiskSpace : ULINT := 0;
    diStatus : DINT := 0;
    diBytesOfMegaByte : DINT := 1048576; // = 1024 * 1024
    diFreeDiskSpaceInMB : LINT := 0;
```

```
END_VAR
```

Program

```
diStatus := FC_GetFreeDiskSpace ( 'usb2msd:1', uliFreeDiskSpace );
IF (diStatus = 0) THEN
    diFreeDiskSpaceInMB := ULINT_TO_LINT(uliFreeDiskSpace) / diBytesOf
MegaByte;
END_IF
```

Result

diFreeDiskSpaceInMB contains the free memory space of the USB memory medium in Mb.

FC_GetTotalDiskSpace: Reads Out the Size of a Memory Medium

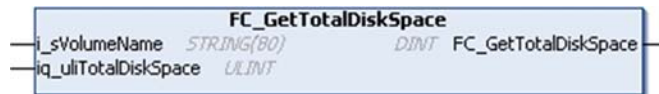
Function Description

The FC_GetTotalDiskSpace function reads out the size of a memory medium (SD card, RAM disk, USB device) in bytes. The name of the memory medium is transferred:

- SD card = ide0:
- RAM disk = ram0:
- USB device = usb2msd:0 (without partition table) or usb2msd:1 (with partition table).

The size of a remote device cannot be read out. If a remote device is specified as parameter, then the function returns -1.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation*.

I/O Variable Description

The following table describes the input variable:

Input	Type	Comment
i_sVolumeName	STRING [80]	Name of the memory device whose memory size shall be read out.

The following table describes the output variable:

Output	Type	Comment
FC_GetTotalDiskSpace	DINT	See the return value description table below.

The following table describes the return values:

Value	Description
0	Free memory space was read out successfully.
-1	Error detected when reading the size.
-318	At least one of the parameters is invalid.

The following table describes the input/output variables:

Input/Output	Type	Comment
iq_uliTotalDiskSpace	ULINT	Size of the memory medium in byte.

Example

Requirement:

The size of the SD card in the controller shall to be determined in Mb.

Declaration

```
PROGRAM RamDiskSize
```

```
VAR
```

```
    uliTotalDiskSpace : ULINT := 0;
    diStatus : DINT := 0;
    diBytesOfMegaByte : DINT := 1048576; // = 1024 * 1024
    diSizeOfRamdiskInMB : LINT := 0;
```

```
VAR
```

Program

```
diStatus := FC_GetTotalDiskSpace ( 'ide0:', uliTotalDiskSpace);
IF (diStatus = 0) THEN
    diSizeOfRamdiskInMB := ULINT_TO_LINT(uliTotalDiskSpace)/diBytesOfMegaByte;
END_IF
```

Result

diSizeOfRamdiskInMB contains the size of the SD card in Mb.

Glossary



A

application

A program including configuration data, symbols, and documentation.

C

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

E

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

I

I/O

(input/output)

P

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.



C

- changing the diagnostic configuration
 - FC_DiagConfigSet2, *15*
- changing the time value for the IEC task cycle time monitoring
 - FC_CycleCheckTimeSet, *25*
- changing the user password
 - FC_UserChangePassword, *58*
- comparing the two logical addresses
 - FC_CompareStLogicalAddress, *31*

D

- data type
 - ST_RetainImageInfo, *9*
- determining the interval time of the task in the IEC program
 - FC_LzsTaskGetCurrentInterval, *26*
- determining the restart status of the controller
 - FC_GetBootState, *53*
- determining the time values for the IEC task cycle time monitoring
 - FC_CycleCheckTimeGet, *24*

E

- enabling/disabling the diagnostic messages for the cycle time overrun
 - FC_CycleCheckSet, *22*
- ensuring that the file system is secure before removing power
 - FC_SysShutdown, *57*

F

- FC_CheckProgramIdent
 - verifying if the memory image fits the application, *42*
- FC_CompareStLogicalAddress
 - comparing the two logical addresses, *31*

- FC_CycleCheckSet
 - enabling/disabling the diagnostic messages for the cycle time overrun, *22*
- FC_CycleCheckTimeGet
 - determining the time values for the IEC task cycle time monitoring, *24*
- FC_CycleCheckTimeSet
 - changing the time value for the IEC task cycle time monitoring, *25*
- FC_DiagConfigRead2
 - reading the diagnostic configuration, *13*
- FC_DiagConfigSet2
 - changing the diagnostic configuration, *15*
- FC_DiagMsgRead
 - reading the diagnostic information, *17*
- FC_DiagQuit
 - resetting the diagnostic message, *20*
- FC_DiagUserMsgWrite
 - triggering the diagnostic message, *19*
- FC_GetBootState
 - determining the restart status of the controller, *53*
- FC_GetFreeDiskSpace
 - reading out the free memory space, *66*
- FC_GetNsPerCPUClockCycle
 - measuring time high-precisely, *61*
- FC_GetNVRamStatus
 - reading the status of NVRam while booting, *52*
- FC_GetRetainImageInfo
 - reading the additional information to be written in the file of a memory image, *44*
- FC_GetTimeOfDay
 - reading the current time of the system, *62*
- FC_GetTotalDiskSpace
 - reading out the size of a memory, *69*
- FC_GetTSC
 - measuring time high-precisely, *64*
- FC_IsStLogicalAddressValid
 - verifying the validation of the logical address, *32*

- FC_LzsTaskGetCurrentInterval
 - determining the interval time of the task in the IEC program, 26
 - FC_LzsTaskGetInterval
 - reading the interval time of the task in the IEC program, 28
 - FC_MsgLogSave
 - storing the message log file on the bulk memory, 33
 - FC_PrgResetAndStart
 - resetting and starting the application using the user program, 29
 - FC_RemoteDeviceCreate
 - setting-up device for file services, 36
 - FC_RemoteUserIdSet
 - setting user name and password for remote file services, 38
 - FC_RetainImageLoad
 - loading the memory image of the retain memory, 46
 - FC_RetainImageSave
 - saving the contents of the retain memory, 49
 - FC_SysReset
 - resetting the controller, 55
 - FC_SysSaveParameter
 - saving custom parameters to the SD Card, 56
 - FC_SysShutdown
 - ensuring that the file system is secure before removing power, 57
 - FC_UserChangePassword
 - changing the user password, 58
- function
- FC_CheckProgramIdent, 42
 - FC_CompareStLogicalAddress, 31
 - FC_CycleCheckSet, 22
 - FC_CycleCheckTimeGet, 24
 - FC_CycleCheckTimeSet, 25
 - FC_DiagConfigRead2, 13
 - FC_DiagConfigSet2, 15
 - FC_DiagMsgRead, 17
 - FC_DiagQuit, 20
 - FC_DiagUserMsgWrite, 19
 - FC_GetBootState, 53
 - FC_GetFreeDiskSpace, 66
 - FC_GetNsPerCPUClockCycle, 61
 - FC_GetNVRamStatus, 52
 - FC_GetRetainImageInfo, 44
 - FC_GetTimeOfDay, 62
 - FC_GetTotalDiskSpace, 69
 - FC_GetTSC, 64
 - FC_IsStLogicalAddressValid, 32
 - FC_LzsTaskGetCurrentInterval, 26
 - FC_LzsTaskGetInterval, 28
 - FC_MsgLogSave, 33
 - FC_PrgResetAndStart, 29
 - FC_RemoteDeviceCreate, 36
 - FC_RemoteUserIdSet, 38
 - FC_RetainImageLoad, 46
 - FC_RetainImageSave, 49
 - FC_SysReset, 55
 - FC_SysSaveParameter, 56
 - FC_SysShutdown, 57
 - FC_UserChangePassword, 58

L

- loading the memory image of the retain memory
 - FC_RetainImageLoad, 46

M

- measuring time high-precisely
 - FC_GetNsPerCPUClockCycle, 61
 - FC_GetTSC, 64

R

- reading out the free memory space
 - FC_GetFreeDiskSpace, *66*
- reading out the size of a memory
 - FC_GetTotalDiskSpace, *69*
- reading the additional information to be written in the file of a memory image
 - FC_GetRetainImageInfo, *44*
- reading the current time of the system
 - FC_GetTimeOfDay, *62*
- reading the diagnostic configuration
 - FC_DiagConfigRead2, *13*
- reading the diagnostic information
 - FC_DiagMsgRead, *17*
- reading the interval time of the task in the IEC program
 - FC_LzsTaskGetInterval, *28*
- reading the status of NVRam while booting
 - FC_GetNVRamStatus, *52*
- resetting and starting the application using the user program
 - FC_PrgResetAndStart, *29*
- resetting the controller
 - FC_SysReset, *55*
- resetting the diagnostic message
 - FC_DiagQuit, *20*

S

- saving custom parameters to the SD Card
 - FC_SysSaveParameter, *56*
- saving the contents of the retain memory
 - FC_RetainImageSave, *49*
- setting user name and password for remote file services
 - FC_RemoteUserIdSet, *38*
- setting-up device for file services
 - FC_RemoteDeviceCreate, *36*
- ST_RetainImageInfo
 - data type, *9*
- storing the message log file on the bulk memory
 - FC_MsgLogSave, *33*

T

- triggering the diagnostic message
 - FC_DiagUserMsgWrite, *19*

V

- verifying if the memory image fits the application
 - FC_CheckProgramIdent, *42*
- verifying the validation of the logical address
 - FC_IsStLogicalAddressValid, *32*

