SoMachine

Getting & Setting Real Time Clock SysTimeRtc and SysTimeCore Library Guide

06/2017

EIO0000000667.08

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.

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.

Table of Contents

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 documentation describes the libraries SysTimeRtc and SysTimeCore.

The SysTimeRtc library implements functions for the real time clock (RTC) management of the runtime system.

The SysTimeCore library provides access to high resolution time routines for time calculations.

The following table shows the list of the functions splitted in the two libraries and the controllers that support them:

Function Name	Library Name	M238	M241, M251, M258 LMC058, LMC078	XBT-GC XBT-GK XBT-GT HMISCU	ATV IMC
SysTimeGetMs	SysTimeCore	x	x	x	x
SysTimeGetNs	SysTimeCore	-	x	x	x
SysTimeGetUs	SysTimeCore	x	x	x	x
SysTimeRtcConvertDateToHighRes	SysTimeRtc	-	x	x	-
SysTimeRtcConvertHighResToDate	SysTimeRtc	-	x	x	-
SysTimeRtcHighResGet	SysTimeRtc	-	x	x	-
SysTimeRtcHighResSet	SysTimeRtc	-	x	x	-
SysTimeRtcControl	SysTimeRtc	-	only supported by LMC078	x	x
SysTimeRtcConvertDateToUtc	SysTimeRtc	x	x	x	x
SysTimeRtcConvertUtcToDate	SysTimeRtc	x	x	x	x
SysTimeRtcConvertLocalToUtc	SysTimeRtc	-	-	-	-
SysTimeRtcConvertUtcToLocal	SysTimeRtc	-	-	-	-
SysTimeRtcGet	SysTimeRtc	x	x	x	x
SysTimeRtcSet	SysTimeRtc	x	x	x	x
SysTimeRtcConvertHighResToLocal	SysTimeRtc	-	-	-	-
SysTimeRtcConvertLocalToHighRes	SysTimeRtc	-	-	-	-
SysTimeRtcGetTimezone	SysTimeRtc	x	x	x	x

x Supported
- Not supported

Function Name	Library Name	M238	M241, M251, M258 LMC058, LMC078	XBT-GC XBT-GK XBT-GT HMISCU	ATV IMC
SysTimeRtcSetTimezone	SysTimeRtc	x	x	x	x
x Supported **-** Not supported					

Validity Note

This document has been updated for the release of SoMachine V4.3.

Related Documents

Document title	Reference
Modicon M238 Logic Controller - System Functions and Variables - PLCSystem Library Guide	*EIO0000000364 (ENG)*; *EIO0000000757 (FRE)*; *EIO0000000758 (GER)*; *EIO0000000759 (SPA)*; *EIO0000000760 (ITA)*; *EIO0000000761 (CHS)*
Modicon M241 Logic Controller - System Functions and Variables - PLCSystem Library Guide	*EIO0000001438 (ENG)*; *EIO0000001439 (FRE)*; *EIO0000001440 (GER)*; *EIO0000001441 (SPA)*; *EIO0000001442 (ITA)*; *EIO0000001443 (CHS)*
Modicon M251 Logic Controller - System Functions and Variables - PLCSystem Library Guide	*EIO0000001468 (ENG)*; *EIO0000001469 (FRE)*; *EIO0000001470 (GER)*; *EIO0000001471 (SPA)*; *EIO0000001472 (ITA)*; *EIO0000001473 (CHS)*
Modicon M258 Logic Controller - System Functions and Variables - PLCSystem Library Guide	*EIO0000000584 (ENG)*; *EIO0000000585 (FRE)*; *EIO0000000586 (GER)*; *EIO0000000587 (ITA)*; *EIO0000000588 (SPA)*; *EIO0000000589 (CHS)*
Altivar ATV IMC Drive Controller - System Functions and Variables - PLCSystem Library Guide	*EIO0000000596 (ENG)*; *EIO0000000597 (FRE)*; *EIO0000000598 (GER)*; *EIO0000000599 (SPA)*; *EIO0000000600 (ITA)*; *EIO0000000601 (CHS)*

Document title	Reference
Modicon LMC058 Motion Controller - System Functions and Variables - PLCSystem Library Guide	EIO0000000566 (ENG); EIO0000000567 (FRE); EIO0000000568 (GER); EIO0000000569 (ITA); EIO0000000570 (SPA); EIO0000000571 (CHS)
Modicon LMC078 Motion Controller - System Functions and Variables - PLCSystem Library Guide	EIO0000001917 (ENG); EIO0000001918 (FRE); EIO0000001919 (GER); EIO0000001920 (SPA); EIO0000001921 (ITA); EIO0000001922 (CHS)
Magelis XBTGC, XBTGT, XBTGK HMI Controller - System Functions and Variables - PLCSystem Library Guide	EIO0000000626 (ENG); EIO0000000627 (FRE); EIO0000000628 (GER); EIO0000000629 (ITA); EIO0000000630 (SPA); EIO0000000631 (CHS)
Magelis SCU HMI Controller - System Functions and Variables - PLCSystem Library Guide	EIO0000001246 (ENG); EIO0000001247 (FRE); EIO0000001248 (GER); EIO0000001249 (SPA); EIO0000001250 (ITA); EIO0000001251 (CHS)

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

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
General Information

General Information

Overview

Depending on the runtime system, the real time clock (RTC) of the controller is provided in standard or high resolution time stamp. That is, the time stamp represents the seconds (standard resolution) or milliseconds (high resolution) since January 1st, 1970 00:00:00. It is also known as UNIX format.

The RTC in standard resolution is supported for all SoMachine controllers. Refer to the table *(see page 7)* in the Document Scope chapter to verify for which controller supports the high resolution RTC.

Chapter 2
Getting Started

Overview

This chapter provides essential information to start using the `SysTime` functions.

What Is in This Chapter?

This chapter contains the following topics:

Adding the Libraries

Procedure

To have access to the functions, it is necessary to manually add one or both libraries:

Step	Action
1	Double-click the **Library Manager** node of the controller **Application** node in the **Tools tree**.
2	Click **Add Library**.
3	Browse to **System → SysLibs**, visible when the **Company** filter is set to **System** or **All Company**. **Result:** The System libraries list is displayed.
4	Select the SysTimeRtc or SysTimeCore library and press **OK**. **Result:** The library is added to the **Library Manager** list.

Get the Controller Date and Time

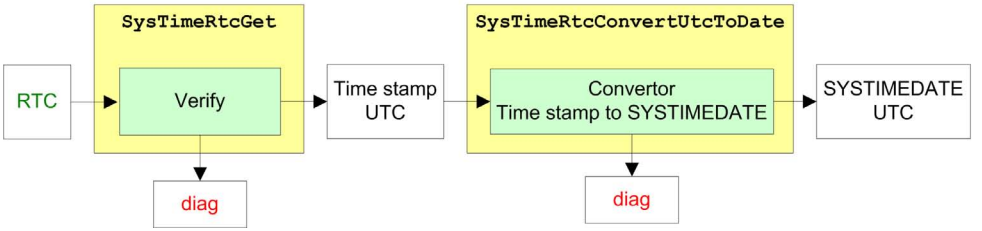
Overview

To get the RTC of the controller in a structured and ergonomic format, you have to use 2 different functions.

1. Read the RTC, using the functions `SysTimeRtcGet` *(see page 28)* or `SysTimeRtcHighResGet` *(see page 29)*.
2. Convert the time stamp in UNIX format to the SYSTIMEDATE format with the use of the function `SysTimeRtcConvertUtcToDate` *(see page 34)* or `SysTimeRtcConvertHighResToDate` *(see page 35)*.

NOTE: Due to the fact that only the UTC (Coordinated Universal Time) time is globally unique, on most systems only the UTC time is stored and processed.

Principle Diagram - Get the RTC of the Controller in Standard Resolution



Example

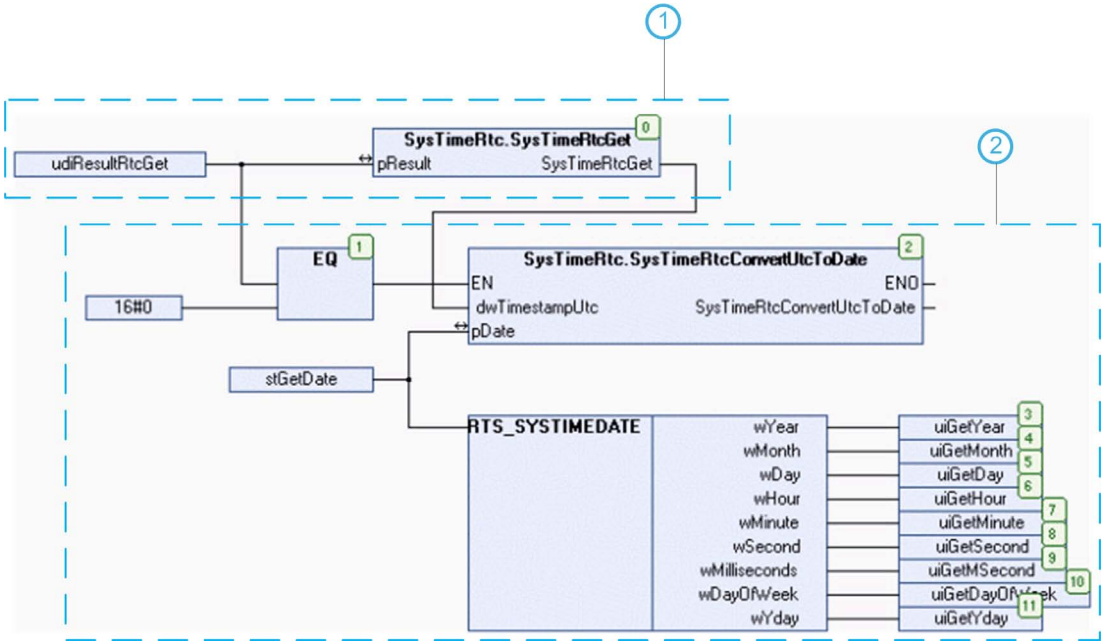
This program example can be used to get the controller date and time.

Variable declaration:

```
VAR
   uidResultRtcGet: UDINT
   stGetDate: SysTimeRtc.RTS_SYSTIMEDATE;
   uiGetYear: UINT;
   uiGetMonth: UINT;
   uiGetDay: UINT;
   uiGetHour: UINT;
   uiGetMinute: UINT;
   uiGetSecond: UINT;
   uiGetMSecond: UINT;
```

```
    uiGetDayOfWeek: UINT;
    uiGetYday: UINT;
    uidResultConvertToDate: UDINT;
END_VAR
```

POU program:



1 Get the RTC of the controller as time stamp value.
2 Convert the time stamp value in SYSTIMEDATE format.

Set the Controller Date and Time

Overview

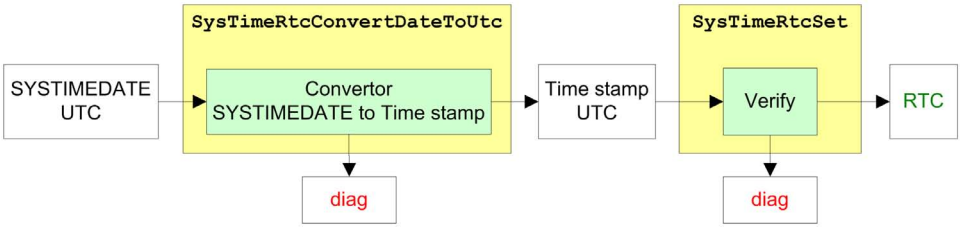
To set the RTC of the controller based on a structured and ergonomic format, you have to use 2 different functions.

1. Convert the SYSTIMEDATE format to the time stamp in UNIX format with the use of the function `SysTimeRtcConvertDateToUtc` *(see page 32)* or `SysTimeRtcConvertDateToHighRes` *(see page 33)*.
2. Write the RTC, using the functions `SysTimeRtcSet` *(see page 30)* or `SysTimeRtcHighResSet` *(see page 31)*.

NOTE: Some controllers support a function for a weekly automatic correction of the real time clock. The name of this function is `SetRTCDrift`. The use of this function could be an alternative to using the `SysTimeRtcSet` function for the continuously readjustment of the RTC. Refer to the PLCSystem Library Guide *(see page 8)* of your controller to verify whether the function is supported and to get further information about this function.

NOTE: Due to the fact that only the UTC (Coordinated Universal Time) time is globally unique, on most systems only the UTC time is stored and processed.

Principle Diagram - Set the RTC of the Controller in Standard Resolution



Example

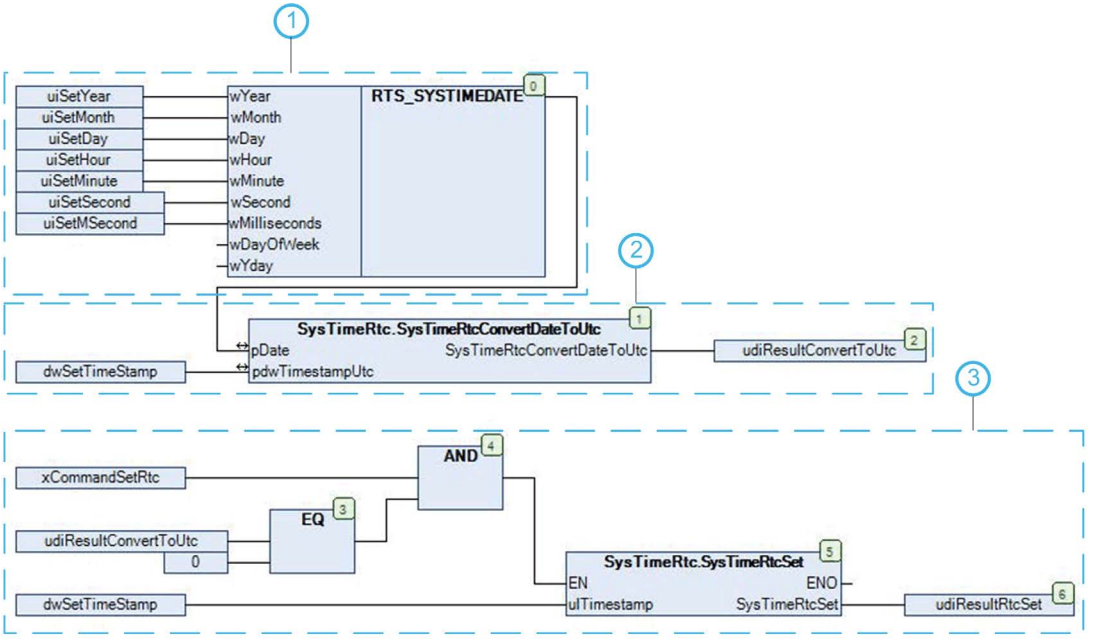
This program example can be used to set the controller real time clock with a user date and time.

Variable declaration:

```
VAR
   uiSetYear: UINT;
   uiSetMonth: UINT;
   uiSetDay: UINT;
   uiSetHour: UINT;
   uiSetMinute: UINT;
```

```
    uiSetSecond: UINT;
    uiSetMSecond: UINT
    udiResultConvertToUtc: UDINT;
    dwSetTimeStamp: DWORD;
    xCommandSetRtc: BOOL;
    uidResultRtcSet: UDINT;
END_VAR
```

POU program:



1 Assign the date and time parameter to the structure.
2 Convert the SYSTIMEDATE format to a time stamp value.
3 Set the controller RTC with new time stamp if `xCommandSetRtc` = TRUE and conversion was successful.

Chapter 3
SysTimeCore Library

Section 3.1
Functions Descriptions

Overview

This section describes the functions of the SysTimeCore library.

What Is in This Section?

This section contains the following topics:

SysTimeGetMs

Function Description

Returns a monotonic rising millisecond counter. This value can be used for timeout and relative time measurements. The counter is reset at each boot-up of the controller.

NOTE: The real time clock does not influence this counter.

Graphical Representation

I/O Variables Description

Output	Type	Description
SysTimeGetMs	UDINT	Value of the millisecond counter, which is reset on boot-up.

SysTimeGetNs

Function Description

Returns a monotonic rising nanosecond counter. This value can be used for timeout and high resolution time measurements.

The counter is reset at each boot-up of the controller.

NOTE: The real time clock does not influence this counter.

Graphical Representation

I/O Variables Description

Output	Type	Description
SysTimeGetNs	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

Input/Output	Type	Description
pUsTime	SYSTIME	Value of the nanosecond counter, which is reset on boot-up.

NOTE: SYSTIME is an alias type based on the data type ULINT.

SysTimeGetUs

Function Description

Returns a monotonic rising microsecond counter. This value can be used for timeout and high resolution time measurements.

The counter is reset at each boot-up of the controller.

NOTE: The real time clock does not influence this counter.

Graphical Representation

I/O Variables Description

Output	Type	Description
SysTimeGetUs	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

Input/Output	Type	Description
pUsTime	SYSTIME	Value of the microsecond counter, which is reset on boot-up.

NOTE: SYSTIME is an alias type based on the data type ULINT.

Chapter 4
SysTimeRtc Library

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
Functions Descriptions

Overview

This section describes the functions of the SysTimeRtc library.

The function for the real time clock handle two different time types:

UTC time: coordinated universal time; has replaced the Greenwich Mean Time. The time zones are given as positive or negative deviation from UTC. For example, UTC+1 corresponds to the Central European Time (CET) and UTC+2 corresponds to the Central European Summer Time (CEST).

Local time: Local time, including daylight saving time (summer time) and time zone shift.

NOTE: Due to the fact that only the UTC (Coordinated Universal Time) time is globally unique, on most systems only the UTC time is stored and processed.

What Is in This Section?

This section contains the following topics:

SysTimeRtcControl

Function Description

This function is used to read the hardware status information related to the real time clock (RTC).

Graphical Representation

I/O Variables Description

Input	Type	Description
iControlTag	DINT	0 = to verify the battery of the RTC 1 = to verify the hour format of the RTC

Input/Output	Type	Description
pdiControlResult	DINT	Result of the verification. Hour format verification: 0 = 12-hour format 1 = 24-hour format Battery verification: 0 = indicates that the battery needs to be replaced 1 = battery is ok

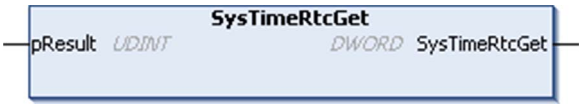
Output	Type	Description
SysTimeRtcControl	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcGet

Function Description

This function is used to read the value of the real time clock (RTC) of the controller. The RTC is provided as time stamp which indicates the number of seconds since January 1st, 1970 00:00:00.

Graphical Representation



I/O Variables Description

Input/Output	Type	Description
pResult	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

Output	Type	Description
SysTimeRtcGet	DWORD	RTC of the controller (number of seconds since January 1st, 1970 00:00:00)

NOTE: An example using this function is provided in this document. *(see page 15)*

SysTimeRtcHighResGet

Function Description

This function is used to read the value of the real time clock (RTC) of the controller. The RTC is provided as high resolution time stamp which indicates the number of milliseconds since January 1st, 1970 00:00:00:000.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pTimestamp	SYSTIME	Time in milliseconds since January 1st, 1970 00:00:00:000

Output	Type	Description
SysTimeRtcHighResGet	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: SYSTIME is an alias type based on the data type ULINT.

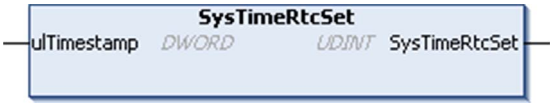
SysTimeRtcSet

Function Description

This function is used to set the real time clock of the controller by a provided time stamp which indicates the numbers of seconds since January 1st, 1970 00:00:00.

NOTE: Setting the RTC of the controller generates entries into the controller log file. Therefore, for automatic adjustments, do not use this function more than once a day.

Graphical Representation



I/O Variables Description

Input	Type	Description
`ulTimestamp`	DWORD	Time stamp value (number of seconds since January 1st, 1970 00:00:00)

Output	Type	Description
`SysTimeRtcSet`	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: An example using this function is provided in this document. *(see page 17)*

SysTimeRtcHighResSet

Function Description

This function is used to set the real time clock (RTC) of the controller by a provided high resolution time stamp value which indicates the number of milliseconds since January 1st, 1970 00:00:00:000.

NOTE: Setting the RTC of the controller generates entries into the controller log file. Therefore, for automatic adjustments, do not use this function more than once a day.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pTimestamp	SYSTIME	Time stamp value (number in milliseconds since January 1st, 1970 00:00:00:000).

Output	Type	Description
SysTimeRtcHighResSet	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcConvertDateToUtc

Function Description

This function converts a date and time in SYSTIMEDATE format *(see page 43)* into the corresponding time stamp value. The time stamp indicates the number of seconds since January 1st, 1970 00:00:00.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pDate	SYSTIMEDATE *(see page 43)*	Date and time to be converted.
pdwTimestampUtc	DWORD	Time stamp calculated from pDate.

Output	Type	Description
SysTimeRtcConvertDateToUtc	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: An example using this function is provided in this document. *(see page 17)*

SysTimeRtcConvertDateToHighRes

Function Description

This function converts a date and time in SYSTIMEDATE *(see page 43)* format into the corresponding high resolution time stamp value. The time stamp indicates the number of milliseconds since January 1st, 1970 00:00:00:000.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pDate	SYSTIMEDATE *(see page 43)*	The time to be converted.
pTimestamp	SYSTIME	Time stamp calculated from pDate.

Output	Type	Description
SysTimeRtcConvertDateToHighRes	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: SYSTIME is an alias type based on the data type ULINT.

SysTimeRtcConvertUtcToDate

Function Description

This function converts a time stamp value into the corresponding date and time in SYSTIMEDATE format. *(see page 43)* The time stamp indicates the number of seconds since January 1st, 1970 00:00:00.

Graphical Representation

I/O Variables Description

Input	Type	Description
dwTimestampUtc	DWORD	Time stamp to be converted.

Input/Output	Type	Description
pDate	SYSTIMEDATE *(see page 43)*	Date and time calculated from input value.

Output	Type	Description
SysTimeRtcConvertUtcToDate	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: An example using this function is provided in this document. *(see page 15)*

SysTimeRtcConvertHighResToDate

Function Description

This function converts a high resolution time stamp value into the corresponding date and time in SYSTIMEDATE *(see page 43)* format. The time stamp indicates the number of milliseconds since January 1st, 1970 00:00:00:000.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pTimestamp	SYSTIME	Time stamp to be converted.
pDate	SYSTIMEDATE *(see page 43)*	Date and time calculated from the input value.

Output	Type	Description
SysTimeRtcConvertHighResToDate	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: SYSTIME is an alias type based on the data type ULINT.

SysTimeRtcGetTimezone

Function Description

This function is used to read the timezone settings of the controller.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pTimezone	TimezoneInformation *(see page 44)*	Timezone settings of the controller.

Output	Type	Description
SysTimeRtcGetTimezone	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcSetTimezone

Function Description

This function is used to set the specified timezone settings.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
pTimezone	TimezoneInformation *(see page 44)*	Timezone settings to be set for the controller.

Output	Type	Description
SysTimeRtcSetTimezone	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcConvertLocalToUtc

Function Description

This function calculates the UTC (Coordinated Universal Time) from a local time considering the timezone setting of the runtime system. Both, the UTC and the local time stamp indicate the number of seconds since January 1st, 1970 00:00:00.

Graphical Representation

I/O Variables Description

Input	Type	Description
dwTimestampLocal	DWORD	Local time stamp

Input/Output	Type	Description
pdwTimestampUTC	DWORD	UTC calculated from the input.

Output	Type	Description
SysTimeRtcConvertLocalToUtc	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcConvertLocalToHighRes

Function Description

This function converts a date and time in SYSTIMEDATE *(see page 43)* format into the corresponding high resolution time stamp value considering the timezone settings of the runtime system.

The value `pDate` represents the local time and will be converted to the UTC value, which is provided on `pTimestamp`. The value `pTimestamp` indicates the number of milliseconds since January 1st, 1970 00:00:00:000.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
`pDate`	SYSTIMEDATE *(see page 43)*	Local time to be converted.
`pTimestamp`	SYSTIME	Time stamp calculated from `pDate`.

Output	Type	Description
`SysTimeRtcConvertLocalToHighRes`	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: SYSTIME is an alias type based on the data type ULINT.

SysTimeRtcConvertUtcToLocal

Function Description

This function calculates the local time from the UTC (Coordinated Universal Time) considering the time zone setting of the runtime system. Both, the UTC and the local time stamp indicate the number of seconds since January 1st, 1970 00:00:00.

Graphical Representation

I/O Variables Description

Input	Type	Description
dwTimestampUtc	DWORD	UTC time stamp

Input/Output	Type	Description
pdwTimestampLocal	DWORD	Local time stamp calculated from the input.

Output	Type	Description
SysTimeRtcConvertUtcToLocal	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

SysTimeRtcConvertHighResToLocal

Function Description

This function converts a high resolution time stamp value into the corresponding date and time in SYSTIMEDATE *(see page 43)* format considering the timezone settings of the runtime system. The value `pTimestamp` represents the UTC and will be converted to the local time, which is provided on `pDate`.

Graphical Representation

I/O Variables Description

Input/Output	Type	Description
`pTimestamp`	SYSTIME	Time stamp to be converted.
`pDate`	SYSTIMEDATE *(see page 43)*	Local time calculated from `pTimestamp`.

Output	Type	Description
`SysTimeRtcConvertHighResToLocal`	UDINT	Runtime system error code (refer to CmpErrors.library): 0 = no error detected

NOTE: SYSTIME is an alias type based on the data type ULINT.

Section 4.2
Data Type Descriptions

What Is in This Section?

This section contains the following topics:

SYSTIMEDATE

Data Type Description

The data type SYSTIMEDATE is an alias type based on the data type RTS_SYSTIMEDATE.

RTS_SYSTIMEDATE

Data Type Description

The data type RTS_SYSTIMEDATE is a structure which implements the elements needed to represent a high resolution date and time value.

This structure contains detailed information on the date and time, presented in a readable format (in contrast to the time stamp).

Name	Type	Description
wYear	UINT	Year
wMonth	UINT	Month, values 1 to12 (1=January, 12=December)
wDay	UINT	Day of month, values 1 to 31
wHour	UINT	Hours of current day, values 0 to 23
wMinute	UINT	Minutes of current hour, values 0 to 59
wSecond	UINT	Seconds of current minute, values 0 to 59
wMilliseconds	UINT	Milliseconds of current second, values 0 to 999
wDayOfWeek	UINT	Day of the week, values 1 to 7 (1=Monday, 7=Sunday)
wYday	UINT	Day of the year, values 1 to 366 (1=January 1st, 365 or 366=December 31st)

TimezoneInformation

Data Type Description

This information describes a local timezone with standard- and daylight-saving-time (also known as summer and winter time).

Name	Type	Description	Unit
`ulStandardDate`	UDINT	Date to switch to standard time (from summer to winter time).	Number of seconds since January 1st, 1970 00:00:00.
`ulDaylightDate`	UDINT	Date to switch to daylight saving time (from winter to summer time).	
`szStandardName`	STRING (32)	Standard name of the timezone (winter time).	String with maximum 32 characters
`szDaylightName`	STRING (32)	Daylight saving time name (summer time).	String with maximum 32 characters
`iBias`	INT	UTC = local time + bias	minutes
`iStandardBias`	INT	Additional offset at standard time (winter time).	minutes
`iDayligthBias`	INT	Additional offset at daylight saving time (summer time).	minutes

Appendices

Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

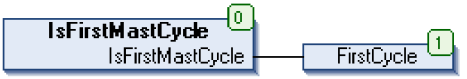
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

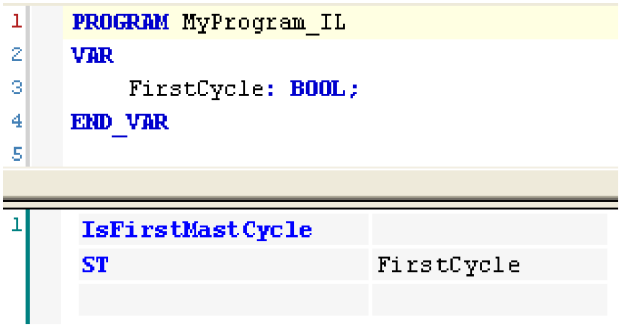
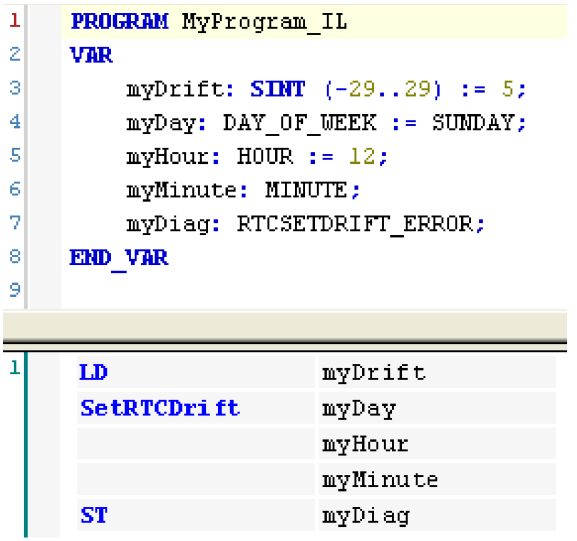
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

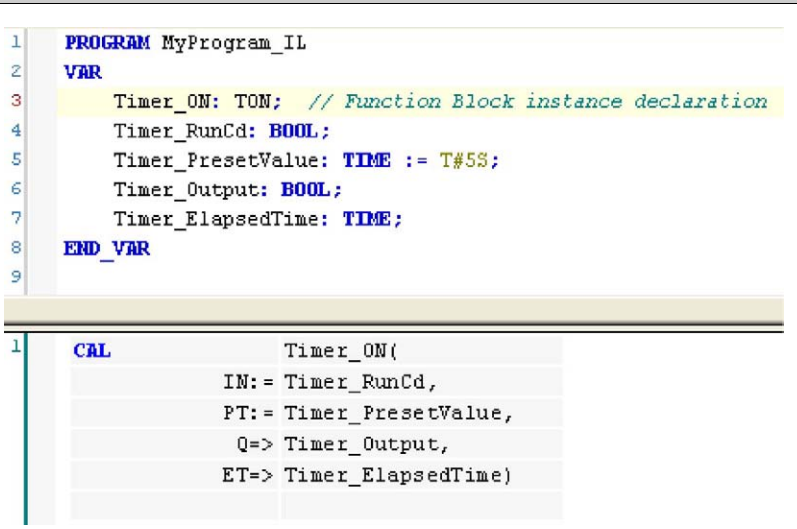
This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the TON Function Block graphically presented below:

Function Block	Graphical Representation
TON	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL             Timer_ON(
          IN:= Timer_RunCd,
          PT:= Timer_PresetValue,
           Q=> Timer_Output,
          ET=> Timer_ElapsedTime)
```

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

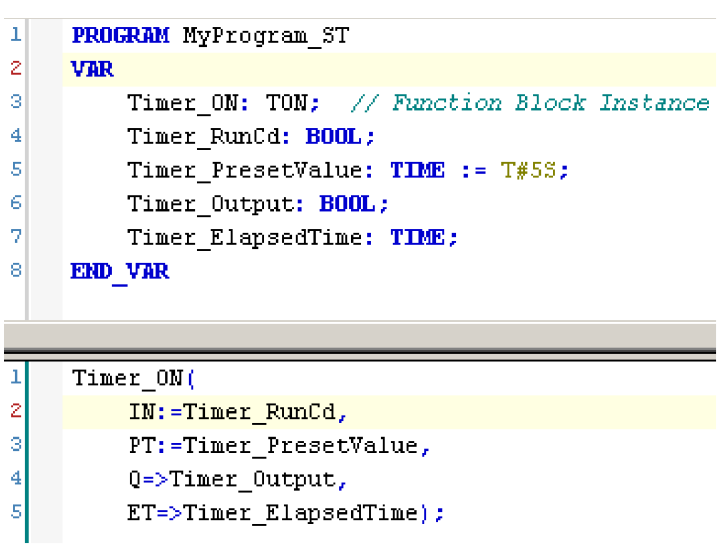
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see SoMachine, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

A

application

A program including configuration data, symbols, and documentation.

B

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

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.

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

I

I/O

(*input/output*)

IL

(*instruction list*) A program written in the language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (refer to IEC 61131-3).

INT

(*integer*) A whole number encoded in 16 bits.

L

LD

(*ladder diagram*) A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (refer to IEC 61131-3).

P

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

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

S

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

U

UTC

(*universal time coordinated*) The primary time standard by which the world regulates clocks and time.

V

variable

A memory unit that is addressed and modified by a program.

Index

S

T

U