SoMachine
Getting & Setting Real Time Clock
SysTime Library Guide

04/2012
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.

© 2012 Schneider Electric. All rights reserved.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Information</td>
<td>5</td>
</tr>
<tr>
<td>About the Book</td>
<td>7</td>
</tr>
<tr>
<td><strong>Chapter 1 Getting Started</strong></td>
<td>11</td>
</tr>
<tr>
<td>Adding the SysTime Library</td>
<td>12</td>
</tr>
<tr>
<td>Get the Controller Date and Time</td>
<td>13</td>
</tr>
<tr>
<td>Set the Controller Date and Time</td>
<td>14</td>
</tr>
<tr>
<td><strong>Chapter 2 Functions Descriptions</strong></td>
<td>15</td>
</tr>
<tr>
<td>2.1 Time Meter Functions</td>
<td>16</td>
</tr>
<tr>
<td>SysTimeGetMs: Returns Time since Power ON in Milliseconds</td>
<td>17</td>
</tr>
<tr>
<td>SysTimeGetUs: Returns Time since Power ON in Microseconds</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Standard Real Time Clock Functions</td>
<td>19</td>
</tr>
<tr>
<td>SysTimeRtcGet: Returns Current Controller Real Time Clock</td>
<td>20</td>
</tr>
<tr>
<td>SysTimeRtcSet: Sets the Controller Real Time Clock</td>
<td>21</td>
</tr>
<tr>
<td>SysTimeRtcConvertUtcToDate: Converts Timestamp into Date &amp; Time</td>
<td>22</td>
</tr>
<tr>
<td>SysTimeRtcConvertDateToUtc: Converts Date &amp; Time into Timestamp</td>
<td>23</td>
</tr>
<tr>
<td><strong>Chapter 3 SYSTIMEDATE: Date &amp; Time Type</strong></td>
<td>25</td>
</tr>
<tr>
<td>SYSTIMEDATE: Date &amp; Time Type</td>
<td>25</td>
</tr>
<tr>
<td><strong>Chapter 4 Implementation Example</strong></td>
<td>27</td>
</tr>
<tr>
<td>Get the Controller Date &amp; Time</td>
<td>28</td>
</tr>
<tr>
<td>Set the Controller Date &amp; Time</td>
<td>29</td>
</tr>
<tr>
<td><strong>Appendices</strong></td>
<td>31</td>
</tr>
<tr>
<td>Appendix A Function and Function Block...</td>
<td>33</td>
</tr>
<tr>
<td>Differences Between a Function and a Function Block</td>
<td>34</td>
</tr>
<tr>
<td>How to Use a Function or a Function Block in IL Language</td>
<td>35</td>
</tr>
<tr>
<td>How to Use a Function or a Function Block in ST Language</td>
<td>38</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td>41</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td></td>
</tr>
</tbody>
</table>
Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, 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 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 an imminently hazardous situation which, if not avoided, will result in death or serious injury.

⚠️ WARNING

WARNING indicates a potentially hazardous situation which, if not avoided, can result in death or serious 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 SysTime functions used to get and set the real time clock, supported by all SoMachine controllers.

The Real time Clock Management is based on functions available in the SysTime library of SoMachine.

The following table shows the list of SysTime library functions and the controllers that support them:

<table>
<thead>
<tr>
<th>Function Name</th>
<th>M238</th>
<th>M258 LMC058</th>
<th>XBT-GC</th>
<th>XBT-GK</th>
<th>XBT-GT</th>
<th>ATV-IMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeGetMs</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SysTimeGetNs</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeGetUs</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertDateToHighRes</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertHighResToDate</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcHighResGet</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcHighResSet</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcControl</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertDateToUtc</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertLocalToUtc</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertUtcToDate</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcConvertUtcToLocal</td>
<td>-</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Function Name

<table>
<thead>
<tr>
<th>Function Name</th>
<th>M238</th>
<th>M258</th>
<th>LMC058</th>
<th>XBT-GC</th>
<th>XBT-GK</th>
<th>XBT-GT</th>
<th>ATV-IMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeRtcGet</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcGetTimezone</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcSet</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>SysTimeRtcSetTimezone</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- Supported
- Not supported

To get information about functions not described in this document, refer to the SysTime_V3x_E document. (See Related Documents)

### Validity Note

This document has been updated with the release of SoMachine V3.1.

### Related Documents

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Location / Part Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTime_V3x_E</td>
<td>On your local SoMachine installation. By default:</td>
</tr>
<tr>
<td></td>
<td>English version Program</td>
</tr>
<tr>
<td></td>
<td>Files\Schneider Electric\SoMachine\Documentation\en\CoDeSys</td>
</tr>
<tr>
<td></td>
<td>German version Program</td>
</tr>
<tr>
<td></td>
<td>Files\Schneider Electric\SoMachine\Documentation\de\CoDeSys</td>
</tr>
</tbody>
</table>
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.


⚠️ 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.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.
Getting Started

Overview

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

What’s in this Chapter?

This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding the SysTime Library</td>
<td>12</td>
</tr>
<tr>
<td>Get the Controller Date and Time</td>
<td>13</td>
</tr>
<tr>
<td>Set the Controller Date and Time</td>
<td>14</td>
</tr>
</tbody>
</table>
Adding the SysTime Library

Procedure

To have access to the SysTime functions, it is necessary to manually add the library:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double-click the Library Manager node of the controller Application node in the Devices window.</td>
</tr>
</tbody>
</table>
| 2    | Click Add Library  
Result: The Add Library dialog box opens. |
| 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 SysTime library and press OK  
Result: The library is added to the Library Manager list. |
Get the Controller Date and Time

Overview

To get the controller Date and Time in \texttt{SYSTIMEDATE} format, it is necessary to use 2 different functions; \texttt{SysTimeRtcGet} (see page 20) and \texttt{SysTimeRtcConvertUtcToDate} (see page 22).

\texttt{SysTimeRtcGet}: This function returns the controller Real Time Clock (RTC) in a timestamp value (number of seconds since January 1st, 1970 00:00:00).

\texttt{SysTimeRtcConvertUtcToDate}: This function converts the Timestamp value into \texttt{SYSTIMEDATE} structure type data providing Year, Month, Day, Hour, Minute, Second values.

Principle Diagram

An implementation example of this principle is available in this document. (see page 28)

Timezone Management

Some controllers (see page 7) support time zone information (summer time shift and Universal Time Coordinated (UTC) shift):

- when this information is not used or set to 0 (default value), the timestamp value returned by \texttt{SysTimeRtcGet} function = RTC.
- when time zone information is not 0, the function returns UTC = RTC + Time Shift.

The Time Shift can be read or set using the Time Zone management functions (\texttt{SysTimeRtcGetTimezone, SysTimeRtcSetTimezone})

UTC timestamp can be converted to local timestamp through \texttt{SysTimeConvertUtcToLocal} function.

\textbf{NOTE}: These functions are not described in this document, refer to the \texttt{SysTime\_V3x\_E} document for more information (see Related Documents (see page 8)).
Set the Controller Date and Time

Overview

To set the controller RTC from Date and Time in SYSTIMEDATE format, it is necessary to use 2 different functions; SysTimeRtcConvertDateToUtc (see page 23) and SysTimeRtcSet (see page 21):

SysTimeRtcConvertDateToUtc: This function converts a SYSTIMEDATE type input into a timestamp value (number of seconds since January 1st, 1970 00:00:00).

SysTimeRtcSet: This function sets the controller RTC with the timestamp input.

NOTE: Check the SetRTCDrift (See your controller PLCSystem Library Guide to check whether the SetRTCDrift function is supported and to get further information about this function) for a weekly automatic correction of the Real Time Clock, as an alternative to using the SysTimeRtcSet function.

Principle Diagram

SET Controller RTC

An implementation example of this principle is available in this document. (see page 29)
Functions Descriptions

Overview

This chapter describes the SysTime functions used to get and set the controller Real Time Clock.

What's in this Chapter?

This chapter contains the following sections:

<table>
<thead>
<tr>
<th>Section</th>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Time Meter Functions</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Standard Real Time Clock Functions</td>
<td>19</td>
</tr>
</tbody>
</table>
2.1 Time Meter Functions

Overview

This section describes the functions for getting time meters (milliseconds and microseconds).

What’s in this Section?

This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeGetMs: Returns Time since Power ON in Milliseconds</td>
<td>17</td>
</tr>
<tr>
<td>SysTimeGetUs: Returns Time since Power ON in Microseconds</td>
<td>18</td>
</tr>
</tbody>
</table>
SysTimeGetMs: Returns Time since Power ON in Milliseconds

Function Description
This function returns the time elapsed since power ON in milliseconds (ms).

Graphical Representation

I/O Variables Description
The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeGetMs</td>
<td>UDINT</td>
<td>Time elapsed since power ON in milliseconds.</td>
</tr>
</tbody>
</table>
Function Descriptions

**SysTimeGetUs: Returns Time since Power ON in Microseconds**

**Function Description**

This function returns the time elapsed since power ON in microseconds (µs).

**Graphical Representation**

![Graphical Representation](image)

**I/O Variables Description**

The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeGetUs</td>
<td>UDINT</td>
<td>Function operation diagnostic: 0 = No error detected 2 = Error detected on input variable, the returned value (pUsTime) is not valid.</td>
</tr>
</tbody>
</table>

The following table describes the input/output variables:

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pUsTime</td>
<td>SYSTIME</td>
<td>Time elapsed since power ON in microseconds.</td>
</tr>
</tbody>
</table>

**NOTE:** SYSTIME type is a ULINT.
2.2 Standard Real Time Clock Functions

Overview

This section describes the controller RTC management standard functions.

What’s in this Section?

This section contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeRtcGet: Returns Current Controller Real Time Clock</td>
<td>20</td>
</tr>
<tr>
<td>SysTimeRtcSet: Sets the Controller Real Time Clock</td>
<td>21</td>
</tr>
<tr>
<td>SysTimeRtcConvertUtcToDate: Converts Timestamp into Date &amp; Time</td>
<td>22</td>
</tr>
<tr>
<td>SysTimeRtcConvertDateToUtc: Converts Date &amp; Time into Timestamp</td>
<td>23</td>
</tr>
</tbody>
</table>
SysTimeRtcGet: Returns Current Controller Real Time Clock

Function Description
This function returns the controller Real Time Clock (RTC) as a Universal Time Coordinated (UTC) timestamp value (number of seconds since 1970, 1st Jan. 00:00:00).

Graphical Representation

I/O Variables Description
The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeRtcGet</td>
<td>DWORD</td>
<td>Controller RTC as a UTC Timestamp value (number of seconds since January 1st, 1970 00:00:00)</td>
</tr>
</tbody>
</table>

The following table describes the input/output variables:

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pResult</td>
<td>UDINT</td>
<td>Function operation diagnostic: 0 = No error detected 1 = Internal operation error detected, the returned value (SysTimeRtcGet) is not valid (0).</td>
</tr>
</tbody>
</table>

**NOTE:** Note: UTC = RTC + Time Shift *(see page 13)*

**NOTE:** An example using this function is provided in this document. *(see page 28)*
SysTimeRtcSet: Sets the Controller Real Time Clock

Function Description
This function sets the controller Real Time Clock by a provided Timestamp value (number of seconds since January 1st, 1970 00:00:00).

Graphical Representation

I/O Variables Description
The following table describes the input variables:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>uITimestamp</td>
<td>DWORD</td>
<td>Timestamp value (number of seconds since January 1st, 1970 00:00:00)</td>
</tr>
</tbody>
</table>

The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeRtcSet</td>
<td>UDINT</td>
<td>Function operation diagnostic: 0 = No error detected 1 = Internal operation error detected</td>
</tr>
</tbody>
</table>

NOTE: An example using this function is provided in this document. (see page 29)
SysTimeRtcConvertUtcToDate: Converts Timestamp into Date & Time

Function Description
This function converts a timestamp value (number of seconds since 1970, 1st Jan. 00:00:00) into the corresponding Date & Time in SYSTIMEDATE format (see page 25).

Graphical Representation

I/O Variables Description
The following table describes the input variables:

<table>
<thead>
<tr>
<th>Input</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwTimestampUtc</td>
<td>DWORD</td>
<td>Timestamp to be converted (number of seconds since January 1st, 1970 00:00:00)</td>
</tr>
</tbody>
</table>

The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysTimeRtcConvertUtcToDate</td>
<td>UDINT</td>
<td>Function operation diagnostic: 0 = No error detected 2 = Error detected on input variable, the returned value (pDate) is not valid.</td>
</tr>
</tbody>
</table>

The following table describes the input/output variables:

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pDate</td>
<td>SYSTIMEDATE</td>
<td>Date &amp; Time calculated from input value (see page 25)</td>
</tr>
</tbody>
</table>

NOTE: An example using this function is provided in this document. (see page 28)
SysTimeRtcConvertDateToUtc: Converts Date & Time into Timestamp

Function Description

This function converts a Date & Time in SYSTIMEDATE format (see page 25) into the corresponding Timestamp value (number of seconds since January 1st, 1970 00:00:00).

Graphical Representation

I/O Variables Description

The following table describes the output variables:

<table>
<thead>
<tr>
<th>Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| SysTimeRtcConvertDateToUtc  | UDINT  | Function operation diagnostic:  
|                             |        | 0 = No error detected  
|                             |        | 2 = Error detected on input variable, the returned value (pdwTimestampUtc) is not valid. |

The following table describes the input/output variables:

<table>
<thead>
<tr>
<th>Input/Output</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pDate</td>
<td>SYSTIMEDATE(see page 25)</td>
<td>Date &amp; Time to be converted</td>
</tr>
<tr>
<td>pdwTimestampUtc</td>
<td>DWORD</td>
<td>Timestamp calculated from entered Date &amp; Time value = number of seconds since January 1st, 1970 00:00:00 or FFFF FFFF hex in case of unsuccessfull conversion.</td>
</tr>
</tbody>
</table>

NOTE: An example using this function is provided in this document. (see page 29)
SYSTIMEDATE: Date & Time Type

Data Type Description

This structure contains detailed information on the Date & Time, presented in a readable format (in contrast to the timestamp).

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wYear</td>
<td>UINT</td>
<td>Year</td>
</tr>
<tr>
<td>wMonth</td>
<td>UINT</td>
<td>Month, values 1..12 (Jan=1..Dec=12)</td>
</tr>
<tr>
<td>wDay</td>
<td>UINT</td>
<td>Day of month, values 1..31</td>
</tr>
<tr>
<td>wHour</td>
<td>UINT</td>
<td>Hours of current day, values 0..23</td>
</tr>
<tr>
<td>wMinute</td>
<td>UINT</td>
<td>Minutes of current hour, values 0..59</td>
</tr>
<tr>
<td>wSecond</td>
<td>UINT</td>
<td>Seconds of current minute, values 0..59</td>
</tr>
<tr>
<td>wMilliseconds</td>
<td>UINT</td>
<td>Milliseconds of current second, values 0..999</td>
</tr>
<tr>
<td>wDayOfWeek</td>
<td>UINT</td>
<td>Day of the week, values 1..7 (Monday = 1..Sunday=7)</td>
</tr>
<tr>
<td>wYday</td>
<td>UINT</td>
<td>Day of the year, values 1..366 (01.Jan=1..31.Dec=365 resp. 366)</td>
</tr>
</tbody>
</table>
Implementation Example

Overview

This chapter provides implementation examples showing how to get or set the Real Time Clock.

What’s in this Chapter?

This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get the Controller Date &amp; Time</td>
<td>28</td>
</tr>
<tr>
<td>Set the Controller Date &amp; Time</td>
<td>29</td>
</tr>
</tbody>
</table>
Implementation Example

Get the Controller Date & Time

Overview

This program example can be used to get the controller Date & Time.

Variable Declaration

```
VAR
// controller current Date & Time
Ctrl_DateTime: SYSTIMEDATE;
CtrlYear: UINT;
CtrlMonth: UINT;
CtrlDay: UINT;
CtrlHour: UINT;
CtrlMinute: UINT;
CtrlSecond: UINT;
CtrlMSecond: UINT;
CtrlDayOfWeek: UINT;
CtrlYday: UINT;
// SysTimeRtcGet operation diag
GetTimeResult: UDINT;
// SysTimeRtcConvertUtcToDate operation diag
UTCToDate_diag: UDINT;
END_VAR
```

POU Program

Get the controller RTC in UTC timestamp value

Convert the current UTC timestamp value in SYSTIMEDATE format
Set the Controller Date & Time

Overview
This program example can be used to set the controller Real Time Clock with a user Date & Time.

Variable Declaration

VAR

// Date & Time to set controller RTC
MyDateTime: SYSTIMEDATE;
MyYear: UINT;
MyMonth: UINT;
MyDay: UINT;
MyHour: UINT;
MyMinute: UINT;
MySecond: UINT;

// SysTimeRtcConvertDateToUtc operation diag
DatetoUTC_diag: UDINT;

// Date & Time in timestamp format
My.UTC: DWORD;

// Controller RTC setting command
Set_DateTime: BOOL;

// SysTimeRtcSet operation diag
RtcSet_diag: UDINT;
END_VAR
Implementation Example

POU Program

Set new Date & Time values in MyDateTime variable

Convert the new Date & Time in UTC timestamp value for RTC settings

Set the controller RTC with new Date & Time if Set_DateTime = TRUE and timestamp value is valid
Appendices
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’s in this Chapter?

This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differences Between a Function and a Function Block</td>
<td>34</td>
</tr>
<tr>
<td>How to Use a Function or a Function Block in IL Language</td>
<td>35</td>
</tr>
<tr>
<td>How to Use a Function or a Function Block in ST Language</td>
<td>38</td>
</tr>
</tbody>
</table>
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
- is always called through 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

Examples: timers, counters

In the example below, Timer_ON is an instance of the Function Block TON:

```plaintext
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

The following procedure describes how to insert a function in IL language:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open or create a new POU in Instruction List language. <strong>NOTE:</strong> The procedure to create a POU is not detailed here. For more information, refer to the SoMachine global help.</td>
</tr>
<tr>
<td>2</td>
<td>Create the variables that the function requires.</td>
</tr>
<tr>
<td>3</td>
<td>If the function has 1 or more inputs, start loading the first input using LD instruction.</td>
</tr>
<tr>
<td>4</td>
<td>Insert a new line below and: &lt;br&gt;  ● type the name of the function in the operator column (left field), or &lt;br&gt;  ● use the Input Assistant to select the function (select <strong>Insert Box</strong> in context menu).</td>
</tr>
<tr>
<td>5</td>
<td>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.</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 SoMachine POU IL Editor**

<table>
<thead>
<tr>
<th>Function</th>
<th>Representation in SoMachine POU IL Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL example of a function without input parameter: <em>IsFirstMastCycle</em></td>
<td><img src="image1.png" alt="IL example of a function without input parameter: IsFirstMastCycle" /></td>
</tr>
<tr>
<td>IL example of a function with input parameters: <em>SetRTCDrift</em></td>
<td><img src="image2.png" alt="IL example of a function with input parameters: SetRTCDrift" /></td>
</tr>
</tbody>
</table>

**Using a Function Block in IL language**

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open or create a new POU in <strong>Instruction List</strong> language. <strong>NOTE:</strong> The procedure to create a POU is not detailed here. For more information, refer to the SoMachine global help.</td>
</tr>
<tr>
<td>2</td>
<td>Create the variables that the function block requires, including the instance name.</td>
</tr>
</tbody>
</table>
To illustrate the procedure, consider this example with the **TON** Function Block graphically presented below:

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Graphical Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TON</td>
<td><img src="image" alt="TON Function Block Graphical Representation" /></td>
</tr>
</tbody>
</table>

In IL language, the function block name is used directly in the **Operator Column**:  

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Representation in SoMachine POU IL Editor</th>
</tr>
</thead>
</table>
| TON            | ```
PROGRAM MyProgram_IL
VAR
Timer_ON: TON; // Function Block instance declaration
Timer_RunCd: BOOL;
Timer_PresetValue: TIME := T#58;
Timer_Output: BOOL;
Timer_ElapsedTime: TIME;
END_VAR
``` |

To use the **Input Assistant** to select the FB (right-click and select **Insert Box** in context menu).

- Function Blocks are called using a **CAL** instruction:
  - Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in context menu).
  - Automatically, the **CAL** instruction and the necessary I/O are created.

Each parameter (I/O) is an instruction:
- Value 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.
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
The following procedure describes how to insert a function in ST language:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 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 the SoMachine global help. |
| 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:

The ST language of this function is the following:

```plaintext
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);
```
Function and Function Block Representation

Using a Function Block in ST Language

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 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 the SoMachine global help. |
| 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,...  
Output1=>VarOutput1, Output2=>VarOutput2,...); |

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

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Graphical Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>TON</td>
<td><img src="image" alt="TON Graphical Representation" /></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Function Block</th>
<th>Representation in SoMachine POU ST Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>TON</td>
<td><img src="image" alt="ST Language Example" /></td>
</tr>
</tbody>
</table>
Glossary

0-9

% Prefix that identifies internal memory addresses in the controller that are used to store the value of program variables, constants, I/O, and so on.

A

application A SoMachine application consists of a program, configuration data, symbols, and documentation.

ASCII American Standard Code for Information Interchange is a communication protocol for representing alphanumeric characters, notably letters, figures and certain graphic and control characters.

ASIC Application Specific Integrated Circuit. Integrated Circuit designed especially for an application. This Integrated Circuit can be analog, digital or both.
Glossary

C

CANopen CANopen is a communication protocol and device profile specification used by Modicon Logic Controllers.

continuous function chart language (CFC) Continuous Function Chart, in extension to the IEC61131-3 standard, is a graphical programming language based on the Function Block Diagram (FBD) language. However, no networks are used but free positioning of graphic elements, which allows feedback loops.

controller A controller, or programmable logic controller (PLC), or else programmable controller is used for automation of industrial processes.

E

EEPROM Electrically Erasable Programmable Read-Only Memory is a type of non-volatile memory used to store data that must be saved when power is removed.

expansion bus Expansion I/O Modules connect to the Modicon Logic Controller using the expansion bus.

expansion I/O modules Optional Expansion I/O Modules are available to add I/O points to a Modicon Logic Controller. (Not all controller models allow expansion).
F

FG

Frequency Generator

firmware

The firmware is the program embedded in the Modicon Logic Controller.

FreqGen

See FG.

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 operand in 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
- is always called through 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

Examples: timers, counters

function block diagram language

(FBD) A function block diagram describes a function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines. An output of a block may also be connected to an input of another block.
G

G

GRAFCET

GRAFCET is used to represent the functioning of a sequential operation in a structured and graphic form.

This is an analytical method that divides any sequential control system into a series of steps, with which actions, transitions, and conditions are associated.

GVL

global variable list in SoMachine managing global variables that are available in all the POUs of the application

H

HSC

High-Speed Counter

I

IEC 61131-3

- IEC for International Electrotechnical Commission, is a not-for-profit, non-governmental international standards organization that prepares and publishes International Standards for all electrical, electronic and related technologies.
- IEC 61131 is a standard for Programmable Logic Controllers (PLCs).
- IEC 61131-3 deals with PLC programming languages and defines two graphical and two textual programming language standards:
  - Ladder Diagram (LD), graphical
  - Function Block Diagram (FBD), graphical
  - Structured Text (ST), textual
  - Instruction List (IL), textual
  - Sequential Function Chart (SFC), has elements to organize programs for sequential and parallel control processing.
instruction list language
(IL) A program written in instruction list language is composed of a series of instructions executed sequentially by the controller. Each instruction is composed of a line number, an instruction code, and an operand. IEC 61131-3 compliant.

IP 20
(Ingress Protection) The protection classification offered by an enclosure is shown by the letter IP and two digits. The first digit indicates two factors: the protection for persons and for equipment. The second digit the protection against water. For more information see refer to the norm NF EN 60.529

ladder diagram language
(LD) A program written in ladder language is composed of graphical representation of instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller. IEC 61131-3 compliant.

latching input
Incoming pulses are captured and recorded for later examination by the application.

located variable
A located variable has an address. (An unlocated variable does not.)

M

MAST
A master (MAST) task is a processor task that is run through its programming software. The MAST task has two sections:
- **IN**: Inputs are copied to the IN section before execution of the MAST task.
- **OUT**: Outputs are copied to the OUT section after execution of the MAST task.
**Glossary**

**master/slave**
Master/slave is a model for a communication protocol where one device or process has unidirectional control over one or more other devices.

**Modbus**
Communication protocol allowing communication between many devices connected to the same network.

**N**

**NEMA**
National Electrical Manufacturers Association publishes standards for the performance of various classes of electrical enclosures. The NEMA standards cover corrosion resistance, ability to protect from rain and submersion, etc. For IEC member countries, standard IEC 60529 is classifying the ingress protection rating (IP Codes) of enclosures.

**network**
Interconnected devices sharing a common data path and protocol for communication.

**node**
An addressable device on a communications network.

**O**

**OS**
Firmware that can be uploaded/downloaded by the user.
PC  
Personal Computer.

PLCopen  
The PLCopen standard brings efficiency, flexibility, and manufacturer-independence to the automation and control industry through the standardization of tools, libraries, and modular approaches to software programming.

Post configuration  
Files that contain machine dependent parameters:
- machine name
- device name or IP address
- Modbus SL address
- Routing table

POU  
A program organization unit includes a variable declaration in source code and the corresponding instruction set. POUs facilitate the modular reuse of software programs, functions, and function blocks. Once declared, POUs are available to one another. SoMachine programming requires the utilization of POUs.

protocol  
A protocol is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints.

PTO  
Pulse Train Output

PWM  
Pulse Width Modulation
Glossary

R

real-time clock
An option that keeps the time for a limited amount of time even when the controller is not powered.

reflex output
In HSC mode, the HSC counter current value is measured against its configured thresholds to determine the state of these dedicated outputs.

RFID
Radio-Frequency Identification is an automatic identification method, relying on storing and remotely retrieving data using devices called RFID tags or transponders.

RPDO
Process Data Object Reception

RTC
See Real-Time Clock.

RUN
A command that causes the controller to run an application program.

S

scan
A controller scans a program and essentially performs three basic functions. First, it reads inputs and places these values in memory. Next, it executes the application program one instruction at a time and stores results in memory. Finally, it uses the results to update outputs.

sequential function chart language
(SFC) A program written in sequential function chart can be used for processes that can be split into steps. SFC is composed of steps with associated actions, transitions with associated logic condition and directed links between steps and transitions. The SFC standard is defined in IEC 848. IEC 61131-3 compliant.
SMS
The Short Message Service is a standard communication service for telephones (or other devices) that send short text messages over the GSM mobile communications system.

SoMachine
SoMachine is a comprehensive Controller Development System software tool for configuring and programming the Modicon Logic Controller and devices compliant with IEC 61131-3.

STOP
A command that causes the controller to stop running an application program.

structured text language
(St) A program written in structured text is composed of complex statements and nested instructions such as: iteration loops, conditional executions, or functions. IEC 61131-3 compliant.

symbol
a symbol is an elementary controller variable published for HMI terminals

System time
Time which is provided in a device by an internal clock

system variable
Unlocated structure that give information on controller data and diagnostic and allow sending commands to controller.
Glossary

**T**

**task**
A group of sections and subroutines, executed cyclically or periodically for the MAST task, or periodically for the FAST task. A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in consequence.

**threshold output**
Coils that are controlled directly by the HSC according to the settings established during configuration.

**TPDO**
Process Data Object Transmission

**U**

**UTC**
Coordinated Universal Time

**V**

**Variable**
Memory unit that can be addressed and modified by a program.
Index

D
Data Types
SYSTIMEDATE, 25

F
Functions
Differences Between a Function and a Function Block, 34
How to Use a Function or a Function Block in IL Language, 35
How to Use a Function or a Function Block in ST Language, 38
SysTimeGetMs, 17
SysTimeGetUs, 18
SysTimeRtcConvertDateToUtc, 23
SysTimeRtcConvertUtcToDate, 22
SysTimeRtcGet, 20
SysTimeRtcSet, 21

S
SYSTIMEDATE
Data Types, 25
SysTimeGetMs
Functions, 17
SysTimeGetUs
Functions, 18
SysTimeRtcConvertDateToUtc
Functions, 23
SysTimeRtcGet
Functions, 20
SysTimeRtcSet
Functions, 21