# SIMOTION

# Frequently asked Questions

Time synchronization between HMI Master (e.g. WinCC), SIMOTION and HMI Slave (e.g. WinCC flexible)

SIEMENS

We reserve the right to make technical changes to this product.

**Copyright**

# General Notes

**Note**

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are correctly used. These Application Examples do not relieve you of the responsibility in safely and professionally using, installing, operating and servicing equipment. When using these Application Examples, you recognize that Siemens cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice. If there are any deviations between the recommendations provided in these Application Examples and other Siemens publications - e.g. Catalogs - then the contents of the other documents have priority.



## Warranty, liability and support

We do not accept any liability for the information contained in this document.

Any claims against us - based on whatever legal reason - resulting from the use of the examples, information, programs, engineering and performance data etc., described in these Application Examples shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act (“Produkthaftungsgesetz”), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract (“wesentliche Vertragspflichten”). However, claims arising from a breach of a condition which goes to the root of the contract shall be limited to the foreseeable damage which is intrinsic to the contract, unless caused by intent or gross negligence or based on mandatory liability for injury of life, body or health. The above provisions do not imply a change in the burden of proof to your detriment.



For questions regarding this application please contact us at the following e-mail address:

mailto:applications.erlf.aud@siemens.com

## Qualified personnel

In the sense of this documentation, qualified personnel are those who are knowledgeable and qualified to mount/install, commission, operate and

service/maintain the products which are to be used. He or she must have the appropriate qualifications to carry out these activities
e.g.:
**Trained and authorized to energize and de-energize, ground and tag circuits and equipment according to applicable safety standards.**

**Trained or instructed according to the latest safety standards regarding the maintenance and use of the appropriate safety equipment.**

**Trained in rendering first aid.**
There is no explicit warning information in this documentation. However, reference is made to warning information in the Operating Instructions for the particular product.

**Reference regarding export codes**

AL: N

ECCN: N

# Contents

Time synchronization FAQ-26234196

# 1 Question

How to synchronize the time of HMI Master (e.g. WinCC), SIMOTION and HMI Slave (e.g. WinCC flexible) ?

# 2 Solution

Time synchronization between WinCC and SIMOTION and between SIMOTION and WinCC flexible is described in the following using an applicative example.

Time synchronization between WinCC and SIMOTION cannot be automatically implemented via the WinCC “Time Synchronization Editor”. The SIMOTION project “timesync“ is therefore required to set the system time from WinCC in SIMOTION. SIMOTION then sets this time on the WinCC flexible panel. The source code is provided in the Appendix to this document.

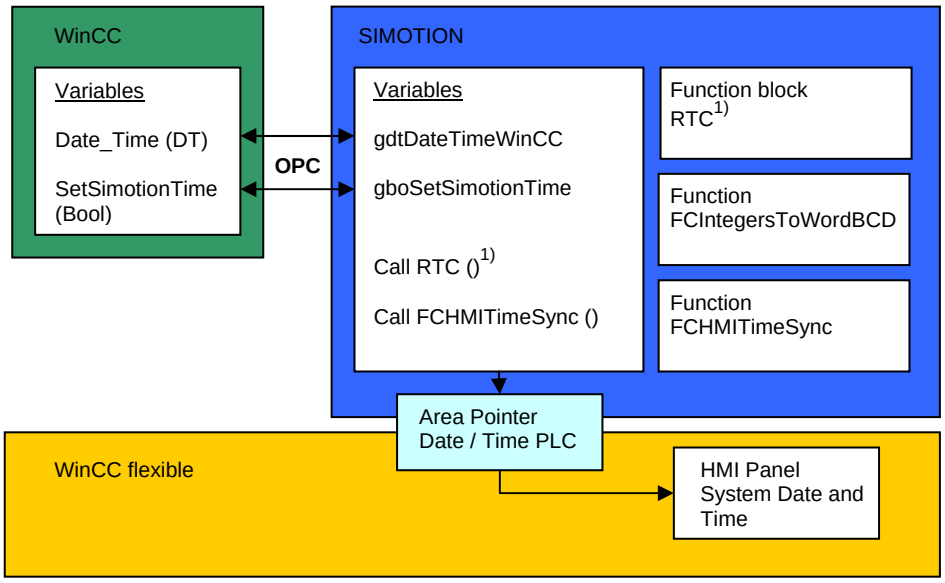
Via OPC, WinCC sets the time in the SIMOTION-CPU (programmable refresh time).

A range pointer is used to synchronize the time between SIMOTION and WinCC flexible.

## 2.1 Project structure

An overview of the project structure is provided in the following figure. The specific input and output parameters of the functions used and their interconnections are also displayed.

Fig. 2-1




1) “Real Time Clock“ SIMOTION function block (see Chapter 2.1.1).

The SIMOTION SCOUT project comprises two ST (Structured Text) units, acting independently of the SIMOTION SCOUT version (see Appendix).

The “TimeSync“ unit includes the “timesync“ program which must be called up with a cyclic task, e.g. the background task. This program sets the WinCC time in SIMOTION and writes the range pointer required to synchronize the time between SIMOTION and WinCC flexible.

The “HMITimeSync“ unit comprises two functions required to generate the range pointer.

**System requirements**

The following program versions have been function-tested:

Table 2-1

| Component | Version |
|---|---|
| SIMATIC Step 7 | V5.4 + SP2 |
| SIMATIC NET PC Software | V6.4 |
| SIMATIC WinCC flexible Advanced | 2005 + SP1 + HF7 |
| SIMATIC WinCC flexible Runtime | 2005 + SP1 + HF7 |
| SIMOTION SCOUT | V4.1.1 |



### 2.1.1 Time synchronization WinCC – SIMOTION

Time synchronization between WinCC and SIMOTION is described in the following.

SIMOTION receives the date, time and “Set“ trigger from WinCC via OPC and sets the time in SIMOTION.

**OPC data communication: WinCC – SIMOTION**

As already mentioned, data communication between SIMOTION and WinCC takes place via the OPC. The SIMATIC NET S7 protocol is required to set up data communication between an OPC client and a SIMOTION controller.

**OPC interface with SIMATIC NET:**

If no SCADA system (WinCC) is available, synchronization between WinCC and SIMOTION can be simulated by means of the additional SIMATIC NET engineering software (subject to license). Variables are exported to OPC tag files using SIMOTION SCOUT and the communication between SIMATIC NET and SIMOTION is tested using OPC SCOUT. When commissioning a production machine equipped with SIMOTION and WinCC flexible at the manufacturer’s premises, this simulation is used to prepare the interface required for convenient Plug&Play connection to the SCADA system WinCC. In most cases, this connection is established after machine delivery when integrating the machine into the plant resp. production line.

**SIMOTION Real Time Clock:**

The SIMOTION system function block “RTC“ is used to set the SIMOTION time. The real time clock (RTC) on the SIMOTION CPU has been defined in the “DATE AND TIME“ type.
The time is set again in SIMOTION after the refresh time parameterized in WinCC (e.g. 10 s). This refresh time can be set by the user.

Table 2-2

| Identifier | Parameter | Data type | Description |
|---|---|---|---|
| SET | Input | BOOL | Set time, FALSE by default |
| READ | Input | BOOL | Read time, FALSE by default |
| PDT | Input | DT | Value to which the real time clock shall be set, DT#0001-01-01-0:0:0 by default.<br>If the value is lower than the preset real time clock value of the SIMOTION unit, the real time clock is set to the preset value (for example, for SIMOTION C: DT#1994-01-01-00:00:00). |
| CDT | Output | DT | Current system time |

The positive SET edge sets the real time clock of the SIMOTION-CPU to the PDT value. If READ is set to TRUE, the current system time is read out and available at output CDT.
The granularity value is 1 ms upon setting the real time clock, the accuracy is defined by the position controller cycle.

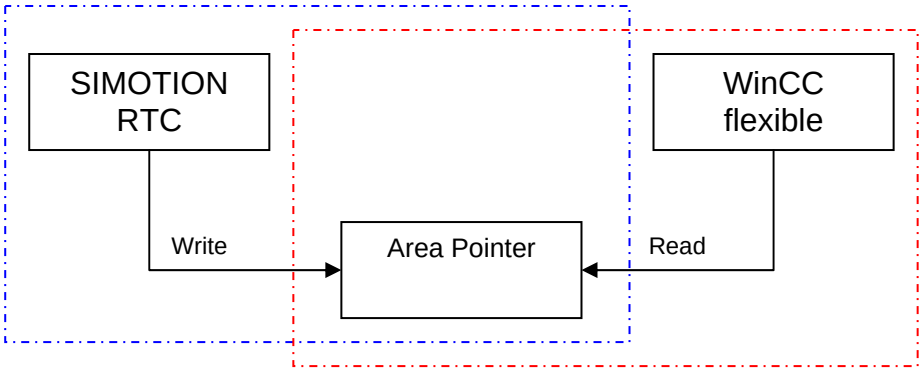
### 2.1.2 Time synchronization: SIMOTION - WinCC flexible

How to synchronize the system time and date of a HMI panel equipped with WinCC flexible with SIMOTION-CPU is described in the following.

A range pointer is required to synchronize WinCC flexible with SIMOTION. The range pointer is used to transfer the date and time from the controller to the operator panel.

The controller loads the range pointer data area via the “FCHMITimeSync“ function. The operator panel cyclically reads in the data using the acquisition cycle configured and synchronizes.

Fig. 2-2




## SIMOTION settings

The range pointer must have the following setup:



Table 2-3

| Data word | High-order byte | | | Low-order byte | | |
|---|---|---|---|---|---|---|
| | 7 | ..... | 0 | 7 | ..... | 0 |
| n+0 | Year (80-99/0-29) | | | Month (1-12) | | |
| n+1 | Day (1-31) | | | Hour (0-23) | | |
| n+2 | Minute (0-59) | | | Second (0-59) | | |
| n+3 | Reserved | | | Reserved | | Weekday (1-7, 1=Sunday) |
| n+4 [1)] | Reserved | | | Reserved | | |
| n+5 [1)] | Reserved | | | Reserved | | |

[1)] The two data words must be provided in the data area to ensure that the data format matches WinCC flexible and avoid reading in incorrect information.

The SIMOTION “FBHMITimeSync“ function block is required to write the range pointer.

**Note** When making entries in the "Year" data area, please ensure that the values 80-99 yield year dates ranging from 1980 to 1999 and the values 0-29 yield year dates ranging from 2000 to 2029.

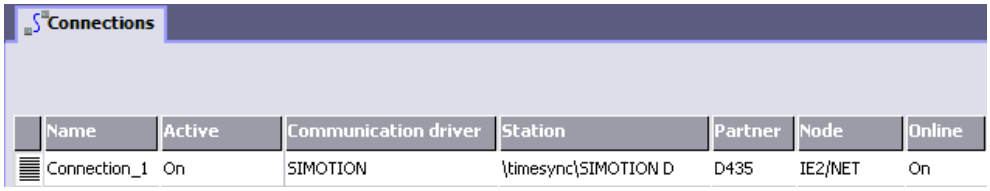
## WinCC flexible settings

The settings required in WinCC flexible for time synchronization are described in the following.

## Communication settings

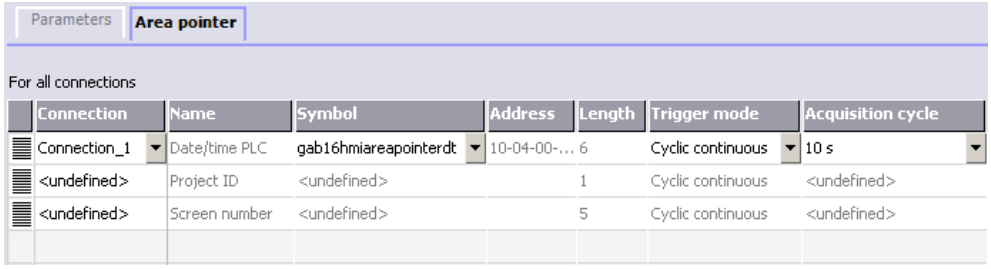
Select Project → Operator Panel → Communication → Connections and accept the settings required.

Fig. 2-3




Select the tabs Range pointer, “for all connections“, “*gab16HMIAreaPointerDT*“ range pointer from the SIMOTION project TimeSync. Then enter the acquisition cycle time required, e.g. 10s.

Fig. 2-4

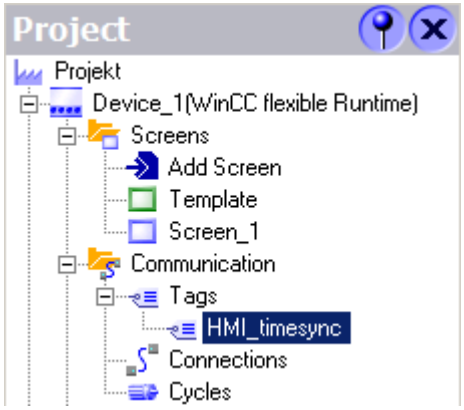



**Note** When configuring, set the range pointer acquisition cycle date/time to a sufficiently high value, since otherwise the operator panel performance may be influenced. Recommendation: Acquisition cycle of 10 s.

### Variable declaration

Insert a new folder called “HMI_timesync“.

Fig. 2-5



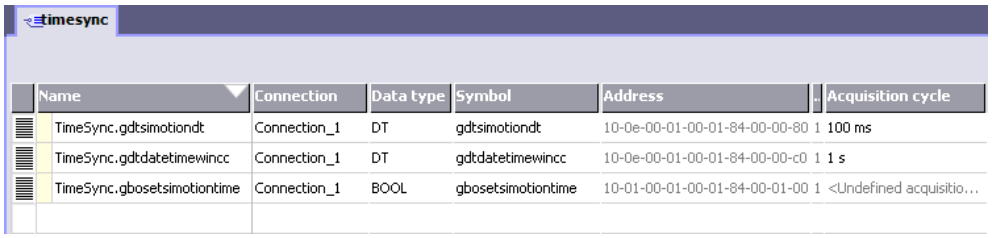

Declare the following variables and enter the acquisition cycle time:

***TimeSync.gdtsimotiondt*** **Acquisition cycle time: 100ms,**

***TimeSync.gdtdatetimewincc*,** **Acquisition cycle time: 1s**

***TimeSync.gbosetsimotiontime*** **Acquisition acc. to request**

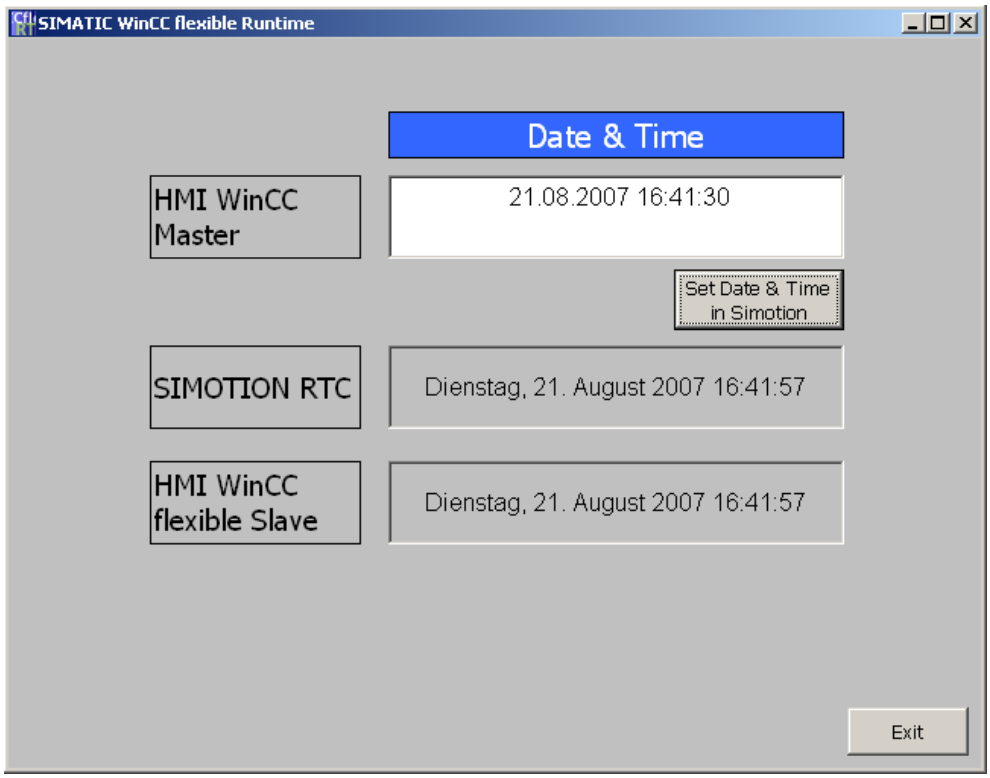
Fig. 2-6



### System time display on WinCC flexible

To display the times (currently synchronized in SIMOTION and on the HMI panel) on the HMI panel (to check the functions upon commissioning), you can generate a picture in WinCC flexible, e.g.:

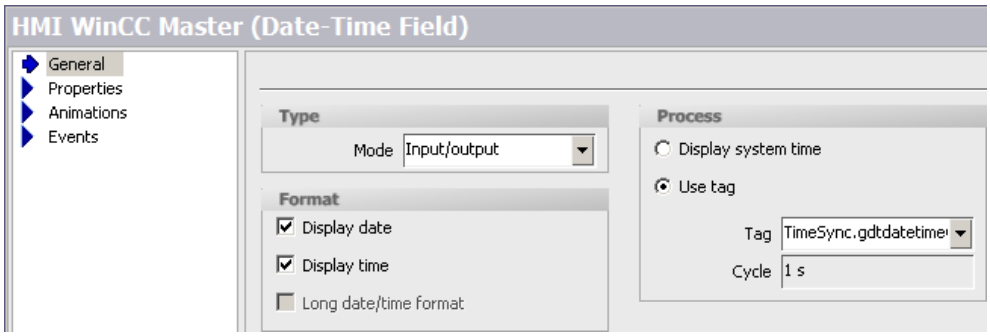
Fig. 2-7




Add a new picture and open it. Under tools, insert three new “Date-Time fields“.

The first “Date-Time field“ (HMI WinCC Master) is used as input field for the time to be set. Please note that this “HMI WinCC Master“ input field simulates the WinCC connection.

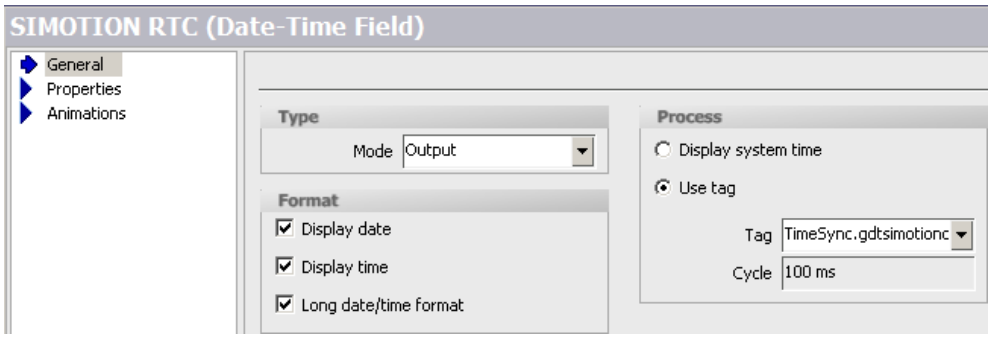
Fig. 2-8




The two following “Date-Time fields“ are used as output fields. The SIMOTION time is displayed in the “SIMOTION RTC“ field and the panel time in the “HMI WinCC flex Slave“ field.

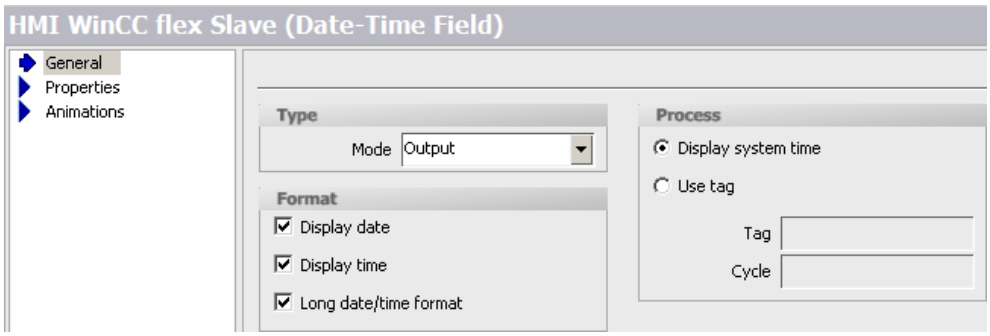
For the “SIMOTION RTC“ field, you have to use the SIMOTION variable *TimeSync.gdtsimotiondt* which has to be declared in WinCC flexible as described above.

Fig. 2-9




We recommend that you select “Long date/time format“ and “Display system time“ for the “HMI WinCC flex Slave“ field. The weekday is also displayed.

Fig. 2-10



The “Set Date & Time in SIMOTION“ button sets the trigger *TimeSync.gbosetsimotiontime*, which sets the time entered in the “HMI WinCC Master“ field in SIMOTION RTC. This trigger simulates the WinCC set trigger.

Fig. 2-11


The times synchronized for HMI panel and SIMOTION are displayed.

**Note** Another possibility to synchronize the time between SIMOTION and WinCC flexible is described in the SIMATIC Customer Support - FAQ article ID 23751258 resp. directly under the following link:
http://support.automation.siemens.com/WW/view/de/23751258

# Appendix

## 3 Project Data

**Project name: timesync**

**Project units: TimeSync, HMItimeSync**

**SIMATIC HMI Station: WinCC flexible RunTime**

### TimeSync source code:

```
//SIEMENS AG
//(c)Copyright 2007 All Rights Reserved
//-----------------------------------------------------------------------------
// file name: TimeSync.st
// library:
// system: SIMOTION
// version: SIMOTION / SCOUT V4.1.1
// restrictions:
// requirements:
// functionality: This unit synchronizes the time of WinCC (HMI Master),
// Simotion and WinCC flexible (HMI slave). OPC is used for the communication
// between WinCC(HMI Master) and SIMOTION. WinCC (HMI Master) writes the date
// and time to a Simotion variable (DT) and sets a trigger, which is used to
// set the Simotion date and time. The Simotion time is cyclically written to
// an area pointer which is read from the slave HMI (WinCC flexible).
// WinCC flexible sets the read time on the slave HMI (WinCC flexible)panel.
//-----------------------------------------------------------------------------
// change log table:
// version    date         expert in charge                changes applied
// 01.00.00   01.08.2007   R.Haro (A&D MC PM APC)          created
//=============================================================================
INTERFACE
//----------- Import ----------------------------------------------------------
    USES HMItimeSync;

// ----------- Device Global Variables -----------------------------------------
    VAR_GLOBAL
        gab16HMIAreaPointerDT : sHMIAreaPointerDTType;
        gdtSimotionDT         : DATE_AND_TIME; // Simotion date and time
        gdtDateTimeWinCC      : DATE_AND_TIME; // date and time from WinCC
        gboSetSimotionTime    : BOOL;          // set Simotion date and time
    END_VAR
// ----------- Export ----------------------------------------------------------
    PROGRAM timesync;
END_INTERFACE

IMPLEMENTATION

// =============================================================================
// ----------- Programs --------------------------------------------------------
// =============================================================================
    PROGRAM TimeSync
//=============================================================================
//(A&D MC PM APC / Erlangen)
//-----------------------------------------------------------------------------
// functionality:
// assignment: Background Task
//-----------------------------------------------------------------------------
// change log table:
// version     date             expert in charge         changes applied
```

```
// 01.00.00    01.08.2007       R.Haro                    created
//============================================================================
        VAR
            dtSimotionRTC : RTC; // Simotion real time clock
        END_VAR

        //Get actual value of real time clock (RTC)
        dtSimotionRTC(set := FALSE, read := TRUE);
        gdtSimotionDT := dtSimotionRTC.cdt;

        //Set Simotion date and time
        IF gboSetSimotionTime = TRUE THEN
            gboSetSimotionTime := FALSE;
            dtSimotionRTC(set := TRUE, read := FALSE, pdt := gdtDateTimeWinCC);
        END_IF;

        // Write date and time to the area pointer
        gab16HMIAreaPointerDT := FCHMITimeSync();

    END_PROGRAM
END_IMPLEMENTATION
```



### HMItimeSync source code:

```
// SIEMENS AG
//(c)Copyright 2007 All Rights Reserved
//----------------------------------------------------------------------------
// file name: HMItimesync.st
// library:
// system: SIMOTION
// version: SIMOTION / SCOUT V4.1
// restrictions:
// requirements:
// functionality: The function "FCHMITimeSync" fills in an area pointer
// used for the time synchronization between SIMOTION and WinCC flexible.
//----------------------------------------------------------------------------
// change log table:
// version    date        expert in charge                changes applied
// 01.00.00   01.08.2007  R.Haro (A&D MC PM APC)          created
//============================================================================
INTERFACE
// ----------- Device Type Definitions ------------------------------------
    TYPE
        sHMIAreaPointerDTType : ARRAY[0..5] OF WORD;
    END_TYPE
// ----------- Export -----------------------------------------------------
    FUNCTION FCHMITimeSync;
END_INTERFACE

IMPLEMENTATION
// ----------- Unit Global Variables --------------------------------------
    VAR_GLOBAL
        dtSimotionRTC : RTC; // Simotion actual date and time
    END_VAR
// ============================================================================
// ----------- Functions --------------------------------------------------
// ============================================================================
    FUNCTION FCIntegersToWordBCD : WORD
// ----------------------------------------------------------------------------
// A&D MC PM APC / Erlangen
// ----------------------------------------------------------------------------
// functionality: This function converts 2 integer variables into 1 Word
// variable in BCD code.
// e.g.
// FB inputs:      +------ input1 -------+   +------ input0 -------+
//                    digit3       digit2        digit1      digit0
```

```
// FB output:      +----------------- outputword ------------------+

// assignment:
// change log table:
// version    date              expert in charge       changes applied
// 01.00.00   01.08.2007        R. Haro                created
// ===============================================================================
        VAR_INPUT
            input0 : USINT;
            input1 : USINT;
        END_VAR

        VAR
            digit0 : WORD;
            digit1 : WORD;
            digit2 : WORD;
            digit3 : WORD;
        END_VAR

        digit0 := USINT_TO_WORD(input0 MOD 10);
        digit1 := USINT_TO_WORD(input0 / 10);
        digit2 := USINT_TO_WORD(input1 MOD 10);
        digit3 := USINT_TO_WORD(input1 / 10);

        FCIntegersToWordBCD := digit0 AND 16#000F
                               OR SHL(in:= digit1 AND 16#000F, n:= 4)
                               OR SHL(in:= digit2 AND 16#000F, n:= 8)
                               OR SHL(in:= digit3 AND 16#000F, n:= 12);
    END_FUNCTION

// ===============================================================================
// ----------- Functions ---------------------------------------------------------
// ===============================================================================
    FUNCTION FCHMITimeSync : sHMIareaPointerDTType
// -------------------------------------------------------------------------------
//A&D MC PM APC / Erlangen
// -------------------------------------------------------------------------------
// functionality: This Function is used to fill in an area pointer with
// the date and time information in order to synchronise the HMI with Simotion
// assignment: (execution level)
// -------------------------------------------------------------------------------
// change log table:
// version     date            expert in charge            changes applied
// 01.00.00    01.08.2007      R.Haro                      created
// ===============================================================================
// ===============================================================================
// The area pointer must have the following format:

//                      Left BYTE                 Right BYTE
//    Data WORD  15   .....   .....   8       7   .....   .....   0
//        n+0             Year                        Month
//        n+1             Day                         Hour
//        n+2             Minute                      Second
//        n+3          *reserved*                   DayOfWeek
//        n+4          *reserved*                   *reserved*
//        n+5          *reserved*                   *reserved*

        VAR
            abDateMarshalling     : ARRAY[0..3] OF BYTE;//Array for marshalling
            todTimeOfDay          : TIME_OF_DAY;  // Simotion time of day
            dDate                 : DATE;         // Simotion date
            tTime                 : TIME;         // Simotion time
            u32JulianDayNumber    : UDINT;        // Gregorian date algorithm
            u32JulianDayNumberRel : UDINT;        // Gregorian date algorithm
            u32DaysCurrentCycle   : UDINT;        // Gregorian date algorithm
            u32DaysSinceGregorianCentury : UDINT; // Gregorian date algorithm
            u32DaysInGreogorianCentury   : UDINT; // Gregorian date algorithm
            u32DaysInJulianYear   : UDINT;        // Gregorian date algorithm
            u32YearsSinceEpoch    : UDINT;        // Gregorian date algorithm
```

```
    u16Date              : UINT;         // Date as UINT value
    u16Year              : UINT;         // Year (yyyy)
    u16Gregorian400Cycles : UINT;        // Gregorian date algorithm
    u16RomanAnnualCylces  : UINT;        // Gregorian date algorithm
    u16Gregorian100Cycles : UINT;        // Gregorian date algorithm
    u16Julian4Cycles      : UINT;        // Gregorian date algorithm
    u8NumberOfMonthsSinceMarch  : USINT; // Gregorian date algorithm
    u8DaysSinceBeginningOfMonth : USINT; // Gregorian date algorithm
    u8Year               : USINT;        // Year usint format (yy)
    u8Month              : USINT;        // Month
    u8Day                : USINT;        // Day
    u8Hour               : USINT;        // Hours
    u8Minute             : USINT;        // Minutes
    u8Second             : USINT;        // Seconds
    u8DayOfWeek          : USINT;        // Day of week
END_VAR

//Get actual value of the real time clock
dtSimotionRTC(set := FALSE, read := TRUE);

//Disassemble DT to TIME_OF_DAY
todTimeOfDay := DATE_AND_TIME_TO_TIME_OF_DAY(dtSimotionRTC.CDT);

//Change the format from TOD to TIME
tTime := todTimeOfDay - TIME_OF_DAY#0:0:0.0;

//Separate seconds out of time
u8Second := UDINT_TO_USINT((tTime / TIME#1s) MOD 60);

//Separate minutes out of time
u8Minute := UDINT_TO_USINT((tTime / TIME#1m) MOD 60);

//Separate hours out of time
u8Hour := UDINT_TO_USINT((tTime / TIME#1h));

//Disassemble DT to DATE
dDate := DT_TO_DATE(dtSimotionRTC.CDT);

//Convert DATE to UDINT by marshalling. u16Date value is the number
//of days since 01.01.1992 AS UDINT. u16Date = 1 in 01.01.1992
abDateMarshalling := ANYTYPE_TO_LITTLEBYTEARRAY(dDate,0);
u16Date := LITTLEBYTEARRAY_TO_ANYTYPE(abDateMarshalling,0);

// Day, month and year will be get using the following algorithm:
// Julian Day Number(JDN)--> Gregorian Calendar (Day/Month/Year).
// In order to use this algorithm, the Julian Day Number is needed.
// It will be calculated by adding the JDN corresponding to the date
// 31.12.1991 to the variable "u16Date".
u32JulianDayNumber := u16Date + 2448622;  // Actual Julian Day

// With "u32JulianDayNumber", compute a relative Julian day
// number "u32JulianDayNumberRel" from a Gregorianepoch. The beginning
// of the Gregorian quadricentennial was 32044 days before the epoch
// of the Julian Period:
u32JulianDayNumberRel := u32JulianDayNumber + 32044;

// With "u32JulianDayNumberRel",compute the number
// "u16Gregorian400Cycles" of Gregorian quadricentennial cycles elapsed
// (there are exactly 146097 days per cycle) since the epoch; subtract
// the days for this number of cycles, it leaves "u32DaysCurrentCycle"
// days since the beginning of the current cycle.
u16Gregorian400Cycles := UDINT_TO_UINT(u32JulianDayNumberRel / 146097);
u32DaysCurrentCycle   := u32JulianDayNumberRel MOD 146097;

// With "u32DaysCurrentCycle",compute the number "u16Gregorian100Cycles"
// (from 0 to 4) of Gregorian centennial cycles (there are exactly
// 36524 days per Gregorian centennial cycle) elapsed since the
// beginning of the current Gregorian quadricentennial cycle, number
// reduced to a maximum of 3 (this reduction occurs for the last day of
```

```
// a leap centennial year where "u16Gregorian100Cycles" would be 4 if it
// were not reduced); subtract the number of days for this number of
// Gregorian centennial cycles, it leaves "u32DaysSinceGregorianCentury"
// days since the beginning of a Gregorian century.
u16Gregorian100Cycles :=
             UDINT_TO_UINT((u32DaysCurrentCycle / 36524 + 1) * 3 / 4);
u32DaysSinceGregorianCentury :=
             u32DaysCurrentCycle - u16Gregorian100Cycles * 36524;

// With "u32DaysSinceGregorianCentury", compute the number
// "u16Julian4Cycles" (from 0 to 24) of Julian quadrennial cycles (there
// are exactly 1461 days in 4 years, except for the last cycle which
// may be incomplete by 1 day) since the beginning of the Gregorian
// century; subtract the number of days for this number of Julian
// cycles, it leaves "u32DaysInGreogorianCentury" days in the Gregorian
// century.
u16Julian4Cycles := UDINT_TO_UINT(u32DaysSinceGregorianCentury / 1461);
u32DaysInGreogorianCentury  := u32DaysSinceGregorianCentury MOD 1461;

// With "u32DaysInGreogorianCentury", compute the number
// "u16RomanAnnualCylces" (from 0 to 4) of Roman annual cycles(there
// are exactly 365 days per Roman annual cycle) since the beginning
// of the Julian quadrennial cycle, number reduced to a maximum of 3
// (this reduction occurs for the leap day, if any, where
// "u16RomanAnnualCylces" would be 4 if it was not reduced); subtract
// the number of days for this number of annual cycles, it leaves
// "u32DaysInJulianYear" days in the Julian year(that begins on March 1)
u16RomanAnnualCylces :=
          UDINT_TO_UINT((u32DaysInGreogorianCentury / 365 + 1) * 3 / 4);

// Convert the four components "u16Gregorian400Cycles",
// "u16Gregorian100Cycles", "u16Julian4Cycles", "u16RomanAnnualCylces"
// into the number "u32YearsSinceEpoch" of years since the epoch,
// by summing their values weighted by the number of years that each
// component represents (respectively 400 years, 100 years, 4 years,
// and 1 year).
u32DaysInJulianYear :=
                u32DaysInGreogorianCentury - u16RomanAnnualCylces * 365;

// With "u32DaysInJulianYear", compute the number
// "u8NumberOfMonthsSinceMarch" (from 0 to 11) of months since March
// (there are exactly 153 days per 5-month cycle, however these 5-month
// cycles are offset by 2 months within the year, i.e. the cycles start
// in May, and so the year starts with an initial fixed number of days
// on March 1, the month can be computed from this cycle by a Euclidian
// division by 5); subtract the number of days for this number of months
// (using the formula above), it leaves u8DaysSinceBeginningOfMonth days
// past since the beginning of the month. You can then deduce the
// Gregorian date (u16Year, u8Month, u8Day) by simple shifts from
// ("u32YearsSinceEpoch", "u8NumberOfMonthsSinceMarch",
// "u8DaysSinceBeginningOfMonth").
u32YearsSinceEpoch := u16Gregorian400Cycles * 400 +
                      u16Gregorian100Cycles * 100 +
                      u16Julian4Cycles * 4 +
                      u16RomanAnnualCylces;
u8NumberOfMonthsSinceMarch :=
            UDINT_TO_USINT((u32DaysInJulianYear * 5 + 308) / 153 - 2);
u8DaysSinceBeginningOfMonth :=
            UDINT_TO_USINT(u32DaysInJulianYear -
                   (u8NumberOfMonthsSinceMarch + 4) * 153 / 5 + 122);

// Get u16Year, u8Month and u8Day:
u16Year := UDINT_TO_UINT(u32YearsSinceEpoch - 4800 +
                       (u8NumberOfMonthsSinceMarch + 2) / 12);
u8Month := (u8NumberOfMonthsSinceMarch + 2) MOD 12 + 1;
u8Day   := u8DaysSinceBeginningOfMonth + 1;

// Transformation from year (yyyy) to year (yy) format.
IF u16Year >= 2000 THEN
```

```
            u8year := UINT_TO_USINT(u16Year MOD 2000);
        ELSE
            u8year := UINT_TO_USINT(u16Year MOD 1900);
        END_IF;

        // Detect day of week, Sunday = 1
        u8DayOfWeek := UINT_TO_USINT(((u16Date + 2) MOD 7) + 1);

        // Array element[0]: Year and Month
        FCHMITimeSync[0] := FCIntegersToWordBCD(input0 := u8Month
                                               ,input1 := u8year);
        // Array element[1]: Day and Hours
        FCHMITimeSync[1] := FCIntegersToWordBCD(input0 := u8Hour
                                               ,input1 := u8Day);
        // Array element[2]: Minutes and Seconds
        FCHMITimeSync[2] := FCIntegersToWordBCD(input0 := u8Second
                                               ,input1 := u8Minute);
        // Array element[3]: Day of week
        FCHMITimeSync[3] := USINT_TO_WORD(u8DayOfWeek);
    END_FUNCTION
END_IMPLEMENTATION
```

# 4 Modifications

Table 4-1: Modifications / Author

| Version | Date / Modification |
|---|---|
| V1.0 | Aug. 1, 2007 / R.Haro |
| | |

# 5 References

## References

This list does not make a claim to be exhaustive and only shows a selection of suitable references.

Table 5-1

| | Topic | Title |
|---|---|---|
| /1/ | | |
| /2/ | | |

# 6 Contact Partners

*Application Center*

SIEMENS

Siemens AG
Automation & Drives
A&D MC PM APC
Frauenauracher Str. 80
Erlangen
Fax: +49 9131-98-1297
mailto: applications.erlf.aud@siemens.com