Programme solution of switching between summer/winter times

Abstract

This application note describes automatic switching between summer/winter time.

Author: Zbyněk Říha
File: ap0022_en_02.pdf

Attachments

File content: –

–	None

Contents

Revision history

Version	Date	Changes
001	3. 12. 2008	New document
002	17. 12. 2009	Name of the function block changed

Related documentation

1. Help tab for DetStudio development environment
   File: DetStudioHelp.chm

1 Automatic change between summer/winter time

Automatic changes between summer/winter times are frequently an individual matter where it is possible to choose from two different solutions in several stages of algorithm design. Half of users are convinced that a solution is correct and the other half prefers another solution. This is why we refrained from our intention to integrate the issue of summer/winter time directly into the operation system or a function module, and we decided to leave the solution up to the author of the application so that they can decide for themselves what solution works the best for them. We offer the following examples as a set of instructions and inspiration.

Let us first state some of the aforementioned issues:

- Regardless of the method you choose (whether you maintain the internal time as UTC and you change the offset in order to calculate local time, or you maintain the internal time as the local time and you keep a flag to distinguish winter time and summer time), you have to maintain a certain complementary information (offset or winter/summer flag) somewhere outside of the real time circuit. That causes problems with application loading (either new or adjusted) when this complementary information does not have a valid value.
- Another problem arises when we want to make sure (and we usually do) that the system switches correctly to summer/winter time even both in case it was operating at the moment of the change and in case it was turned off at the moment of the change.
- Another problem arises in systems where an automatic switch to summer/winter time is supposed to occur, but at the same time there is an option for the user to set the date and time manually. If the user manually changes the date from 15. 3. to 15. 4, should the system switch to summer time immediately, adding one hour? Half users say yes and another half say no. And what if the user changes time from 1:05 AM to 4:05 AM exactly on the Sunday the switch is supposed to occur? Should an hour be added immediately, i.e. set the time moved by an hour against the time the user has just set? Here, everyone probably answers “no”, with the reasoning that the newly set information should be respected and considered to fall in the new time (summer or winter time). In terms of programmes, it is very difficult to distinguish a manual change in time from normal operation of the clock; reading the clock regularly and assessing the switching moment with a subsequent adjustment is not sufficient. It would become necessary to follow up on the event of manual change in time directly in the system and to operate it with a different algorithm than the change caused by the operation of the clock - to circumnavigate the standard periodic time data check. However, DetStudio offers virtually no necessary means to do so.
- Setting time manually poses another problem – it is generally required that the system has a defined solution for a situation when the user enters invalid time data (e.g. 28. 3. 2008, 02:30:00 AM is invalid, because at 02:00 AM the clock moved to 03:00 AM due to switching to summer time, and the time 2:30 AM simply does not exist on this day). Putting in such an invalid time specification is an error on the user's part, but it is an understandable request to prevent this mistake from causing serious problems to the system. However, the way the system is supposed to respond is a matter of personal attitude.
- In the end, we also need to take into account whether the law (or a regulation) defining the algorithm for date of change (last Sunday in March/last Sunday in October) will still be in effect in the long-term perspective or whether a unification of Europe and USA will occur (since in USA, the date of the spring switch is on the second Sunday in March and the date of the autumn switch is on the first Sunday in November).

1.1 Solution example 1

We offer you the following algorithm description which shows one of the solution methods.

We create the programme as follows:

```
//Finding out the current time in the control system
GetTime RTC_time, RTC_items, NONE
```

```
//Converting to the time at the Azores (the summer/winter switch takes place at
midnight)
Let UTC_time = if(@tmSummerTim, RTC_time - 10800, RTC_time - 7200)

//Converting the time from DB-Net format to items
ParseTime UTC_time[0,0], NONE, NONE, UTC_items

//Based on individual items, we find out whether we are in winter or summer time
period
Let @tmSummer = ((UTC_items[4, 0] > 3) and (UTC_items[4, 0] < 10) or (UTC_items[4, 0]
     == 3) and ((UTC_items[3, 0] + 7) - UTC_items[6, 0] > 31)) or (UTC_items[4, 0]
     == 10) and ((UTC_items[3, 0] + 7) - UTC_items[6, 0] <= 31)

//After we load the application, this part is not performed during the first cycle of
this process
If @tmSummerIni, :NONE

     //Learning whether the switch between summer/winter time has occurred
     Let @tmChange = @tmSummer xor @tmSummerTim
     If @tmChange, :NONE

               //If the switch has occurred, we get the current time and shift it
               GetTime RTC_time, NONE, NONE
               Let RTC_time = if(@tmSummer, RTC_time + 3600, RTC_time - 3600)

               //we write the shifted time back into the control system
               SetTime RTC_time

          EndIf

EndIf

//Auxiliary variable for assessing whether the switch between summer/winter time has
occurred
Let @tmSummerTim = @tmSummer

//Auxiliary variable for detection of the first process cycle (after the application
implementation)
Let @tmSummerIni = true
```

The programme must be invoked from a periodic process with a correct period as often as we want specifically the system to “hit” the moment of the switch from summer time to winter time and back. However, the period must not be shorter than 1 second (due to problems with asynchronous writing of time into the hour circuit in the operation system).

The aforementioned algorithm provides the summer/winter time change by maintaining a local time (summer or winter) in the hour circuit and a flag marking this time as summer or winter time is kept in the application database (bit `@tmSummerTim`). This also solves the situation when the current station time is considered to be set correctly based on the previous settings, only a flag marking it as summer or winter time is set. The algorithm handles the issue of manual setting of the switch between summer/winter with the approach – change the time. The issue of manual setting of the switch between summer/winter is not resolved satisfactorily (when attempting to set the date 28. 3. 2008 a new time 4:00 AM at 1:00 AM, the time immediately changes to 5:00 AM because the time automatically switches from the winter time to summer time).

1.2 Solution example 2

This example is a modification to the solution example 1. We add the following lines between the second and the third line of code:

```
If @SysStart, :NONE

Else :NONE

     Let @tmSummerIni = if((UTC_time < LastUTC_t) or (UTC_time > LastUTC_t + 2),
     bool(0), @tmSummerIni)

EndIf

Let LastUTC_t = UTC_time
```

and we add the following line to the subroutine ending:

```
Let @SysStart = False
```

We add the following line into the INIT process (which we create if necessary):

```
Let @SysStart = True
```

Attention
*The underlined **constant = 2** is valued for a one-second period of programme invocation. If we invoke the programme with a longer period, this number must change at least to **Period + 1**; however, we recommend to set the value to **2 × Period**, just to be safe.*

This adjustment provides that if we change the date or time manually, it will not automatically switch to summer/winter time. Only a flag **`@tmSummerTim`** is going to be set according to the set date and time.

The adjustment consists in checking whether the time between two programme invocations has not changed by more than it would be suitable in normal clock operation for the period between process invocations. In such case, it is presumed that there has been a manual change and the lines performing the automatic time switch are skipped, therefore only the flag value **`@tmSummerTim`** gets adjusted to match the newly entered date and time.

1.3 Principle of operation in solutions suggested

Local time converted to zone (=winter) time at the Azores, i.e. time running 1 hour late compared to UTC (GMT) is saved in the second line of the programme into the variable **`UTC_time`**. Time adjusted in this manner has the advantage that the GMT summer/winter switch always takes place at midnight in it, so there is no need to deal with special features of various times within the critical Sunday on which the time switch occurs.

With a time stated in this manner, we are able to find out whether it falls into the summer period or the winter period (bit **`@tmSummer`**) with the following method:

Summer is from April to September; and in March only in cases the closest following Sunday is already to take place in April, and in October only in cases the closest following Sunday is still to take place in October.
With the exception of the moment immediately after the application loading (in example 2 also with the exception of the moment after a manual time change), this flag is compared to the current value of the flag **`@tmSummerTim`**. If they do not correspond, the flag **`@tmChange`** is set and time moves one hour forward or backward.

In any case, the bit **`@tmSummer`** copies itself into the bit **`@tmSummerTim`**.

It is not entirely necessary to implement the variable **`RTC_Time`**, we can replace it with the variable **`UTC_Time`** in all places it had been used. It is useful in the testing state, because then we have both work times separated - the variable **`RTC_Time`** still contains the local time (summer or winter), the variable **`UTC_Time`** still contains the time zone of the Azores – see above.

Attention
The solutions suggested apply exclusively to Central European Time!

1.4 Advantages and disadvantages to both solutions

- The example 2 handles the issue of manual time setting better. The changed time is preserved in the way it had been entered, with no attempts of automatic switch to summer or winter time.
- The example 1 handles the invalid time information better (falling into the omitted hour when switching to summer or winter time) – it immediately changes the time to a valid time by adding or subtracting one hour from it. In such a situation, example 2 leaves the set invalid time until its spontaneous change into valid time (by exceeding 03:00:00) bit **`@tmSummerTim`** oscillates between zero and one.

During the first programme initialization, immediately after the application implementation, there is one small incorrect thing in both examples, because when converting the local time to the Azores time (**`UTC_Time`**) the bit **`@tmSummerTim`** is used that does not have the right value yet at that moment, which may cause an hour shift. The time **`UTC_Time`** is then used to determine the decisive moment for the time switch. This could only cause problems in case a new application was loaded at the moment less than an hour away from the moment of switch to summer/winter time. With respect to very low probability of such an event, we do not consider it necessary to solve this incorrect matter by making the algorithm more complicated. Furthermore, it is practically impossible to solve this situation reliably in case of the autumn switch. At the moment the only information available is the time from the hour circuit, there is no basis for determining whether the time on 31. 10. 2008, 02:30:00 AM is summer time or winter time.

1.5 Function block for summer/winter time switch

A special function block called RTC.sb has been created for the switch to summer/winter time. It is included in the DetStudio installation. The block has been programmed according to example 2 and it is open for any user modifications.

Attention
The function block, written as it is, must be inserted into the process with a period of 1 second!

2 Technical support

All information on the programme solution of switch between summer/winter time will be provided by the technical support department of the company AMiT. Do not hesitate to contact the technical support via e-mail using the following address: **support@amit.cz**.

3 Warning