MANAGER TOOLKIT SAMPLE FOR HISTORIC TREND PERIOD

KEY WORDS: SV32, SV MANAGER, VARIABLES, DLL

EVOLUTIONS

Revision	Writer	Action	Date
1	CBO	Initial version	15/07/20107
2	DL	corrections	09/11/2007
3	CBO	Addition of Dateshift	28/01/2008
4	CBO	Addition of NotifyTriggerVariableName Change of TriggerVariableName	25/03/2008
5	CBO	Activate/Deactivate TRACE in event viewer	13/07/2010

content

Scope of this document

The scope of this document is to describe and explain functional aspects of the program Historic Trend Period. It is targeted at developers willing to understand and develop an interface to SV.

.

Introduction

The object of this project is, via an .ini and .Dll file, capture the value of Scada sofware variables into a data base or trend, at a time defined and in the cyclic period defined.

This sample is composed in two parts:

- Scada project in folder ManagerToolkit\Source\svmgrHisTrendPeriod\ScadaProject
- Visual C++ project in folder ManagerToolkit\Source\svmgrHisTrendPeriod\ BlankManagerProject

"HistoBlankMgr.ini" is file to setup the requirements of capture (Time, variables, etc)

At the startup of the project Scada software run the DLL and setup all the requirements defined in the .ini file. Then the DLL will have the control to update the variables defined.

The file .ini has the following parameters:
[HIS_PERIODTREND1]
AttributeNumber = 4
AttributeValue = BILAN2
TriggerVariableName = His.TrendTrigger
TimeStamped = 10:56:00
Period = 60
Synchronize = 5
Suffix = Other
Dateshift = 0
NotifyTriggerVariableName = His.NotifyTrigger
That will define the variables to update.

INI File: HistoBlankMgr.ini

The 10 elements defined in this file needed for the user DLL are:

AttributeNumber: this notifies to the DLL what attribute of the variables has to match with the text defined in *AttributeValue.*

Format: Number

AttributeValue: this notifies to the DLL what text has to search in the attribute number (defined before) of the variables. The variables that match will be advised.

Format: Text

TriggerVariableName: this notifies to the DLL the variable name that triggers at all the advised variables, when this variable changes from 0 to 1. This variable could be a BIT.
Note1: Nothing happen if the change came from a state NS status.

Format: Text (Scada software variable)

For example:
NS to 1 or NS to 0, nothing will be done. But ONLY 0 to 1 will produce the trigger.

TimeStamped: this is the time to write over the variables. This time will be *rewrited* over the time of all the variables clones (with the suffix defined) produced in reference to the advised variables with the *AttributeValue* defined in the *AttributeNumber* defined**.** This produces the update of these variables into the database (only analog variables).

Format: hh:mm:ss (hours: minutes: seconds)

Period: is the cyclic time in seconds that will be updated all the variables selected

Format: ssss (seconds)

Synchronize: is the minute to synchronize, the possible values are: 1, 5, 15, 30 and 60 seconds.
Deactivate this cyclic task only using -1 number, other number will give an error message in log viewer.

Format: Number (1, 5, 15, 30 and 60)

Suffix: is the suffix to capture the value of Scada sofware variables into a data base or trend in a clone variable desired or the same variable if less empties this option. Moreover if it doesn't exists will be captured over the same variable.

Format: text

For example:

A Variable: *Variables.var1* with attribute number and attribute value defined in .ini file
And a *Suffix = _other* defined in .ini file. The capture of values will be over *Variables.var1_other*
And a *Suffix =* defined in .ini file. The capture of values will be over *Variables.var1 (the same)*

Dateshift: is to save the variable values with a offset of time in days (-1,0 or 1) .

Format: number (in days), only possible values: -1, 0 and 1.

For example:
Dateshift = 1
If the date is January 28, 2008 4:56 and "TimeStamped= 05:00:00" will save to January 29, 2008 05:00:00
If the date is January 31, 2008 4:56 and "TimeStamped= 05:00:00" will save to February 1, 2008 05:00:00

NotifyTriggerVariableName: this notifies to the DLL the variable name of the Notify Trigger Variable. When are updated all the variables with the attribute **AttributeValue**, this variable will be wrote with the value 1 (if a measure) or True (if state).

Format: Text (Scada software variable)

For example:
NotifyTriggerVariableName = NotifyVar
This show a variable called "NotifyVar" will be searched and wrote with the value 1 or True at the moment of update of all the variables with the attribute **AttributeValue.**

ActivateTrace: this option enables or disables the TRACE in the event viewer.

Format: Number (1 or 0)

For example:
ActivateTrace=1, this will enable the message in the event viewer.

There are defaults values if some elements are not defined or if in the .ini file don't exist:
AttributeNumber = 3
AttributeValue = BILAN
TriggerVariableName = triggerbit
TimeStamped = 00:00:00
Period = 60
Synchronize = 1
Suffix =
Dateshift = 0
NotifyTriggerVariableName = NotifyVar
ActivateTrace = 1
Note: Suffix is empty

Example of the file HistoBlankMgr.ini:

[HIS_PERIODTREND1]
AttributeNumber = 4
AttributeValue = BILAN2
TriggerVariableName = triggerbit
TimeStamped = 12:02:00
Period = 60
Synchronize = 1
Suffix = Other
Dateshift = 0

This file defines all the variables of Scada sofware that will advise the DLL with the attribute number "4" wrote "BILAN2", and will write over the clone variables (note that clones variables are without attributes!) with suffix Other

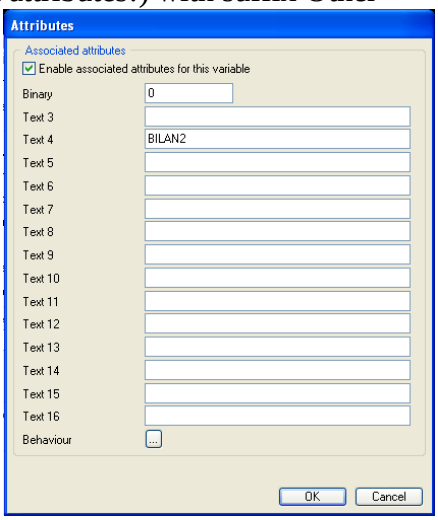


Figure 1: Attribute of variable

If the variable "triggerbit" in SV changes the value to 1 (for example) for a time (and then change to 0 for example), that will update the value and time into the HDS (data base) of all the variables advised with attribute BILAN2 with the actual value and the time of "TimeStamped" in .ini file . That will be twice: when change to 1 and when change to 0. But remember if the value of variables doesn't change, nothing will be updated into the HDS in the second time, because the value and Time is the same.

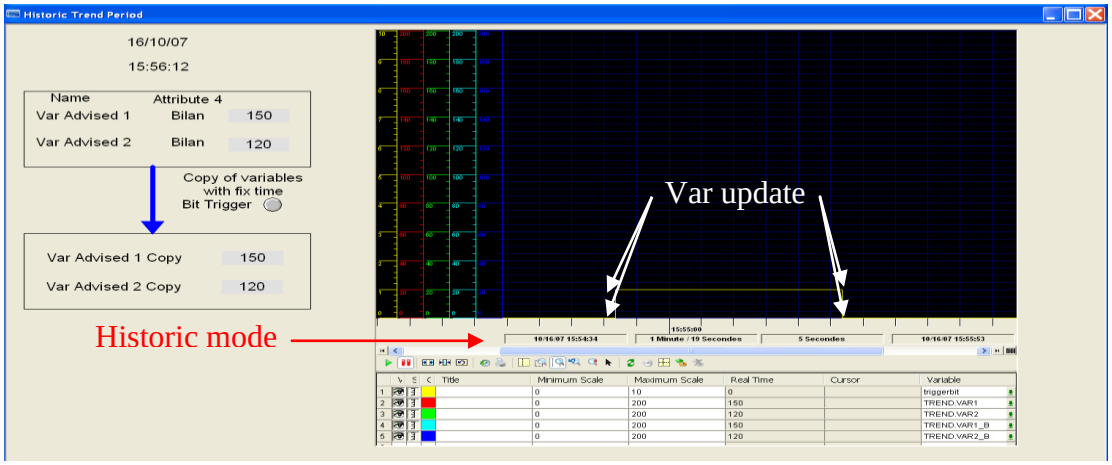


Figure 2: example of trigger a 12:02:00

At the same time will be possible to synchronize all the variables defined (with the attribute defined in .ini file). “Period” and “Synchronize” define this task.

In this example:

Period = 60
Synchronize = 1

Define that each 60 seconds will be updated the data into the data base, and at startup will take 1 second to synchronize.

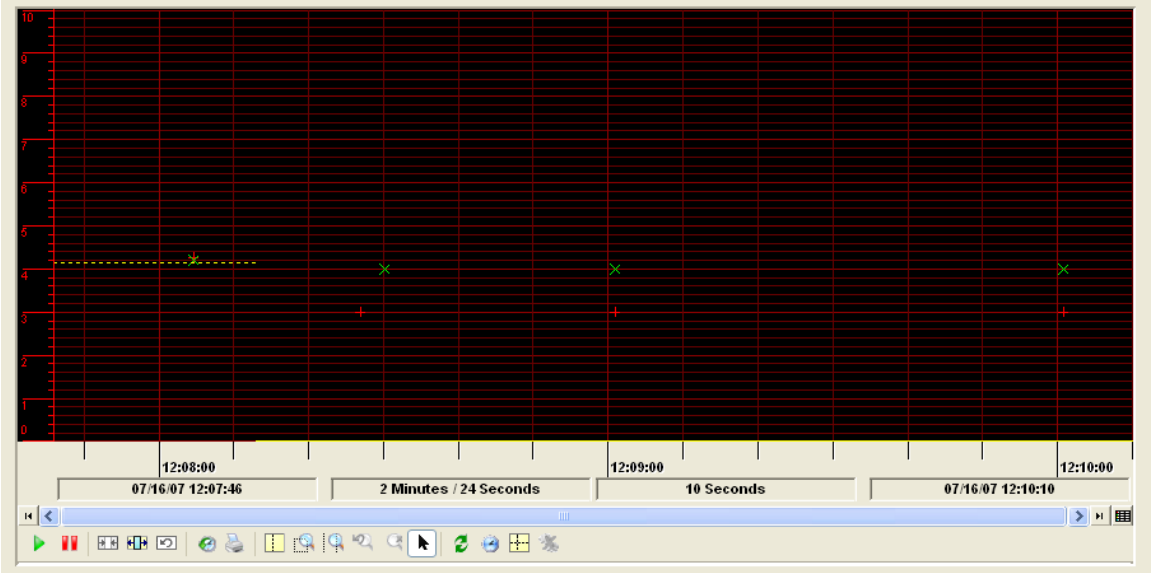


Figure 6: example of cyclic update

Scada project: HisPeriodTrend

This project shows 2 variables that will be updated on the moment specified in HistoBlankMgr.ini. and one variable that trigger.

@TREND.VAR1 **"Var Advised 1"**
@TREND.VAR2 **"Var Advised 2"**
@ triggerbit **"Bit Trigger"**

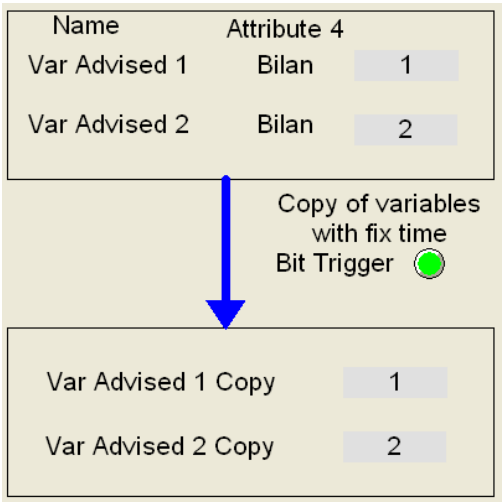
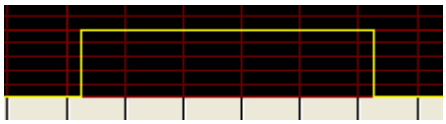


Figure 3: Variables

At the same time, a trend exists to see what happened.
A red cross to the "var advised 1"
A green X to the "var advised 2"
And a yellow line to show the trigger 

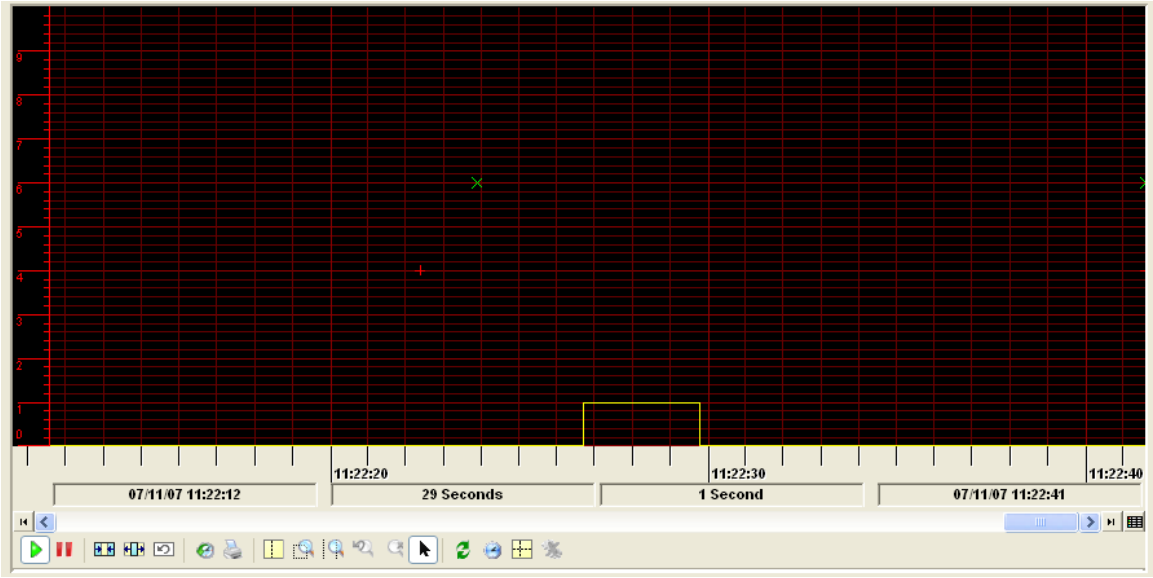


Figure 4: Trend

Is very important remember to setup the attributes defined in the .ini file. It is to advise the variables required to update in the cyclic time that was defined in the .ini file.

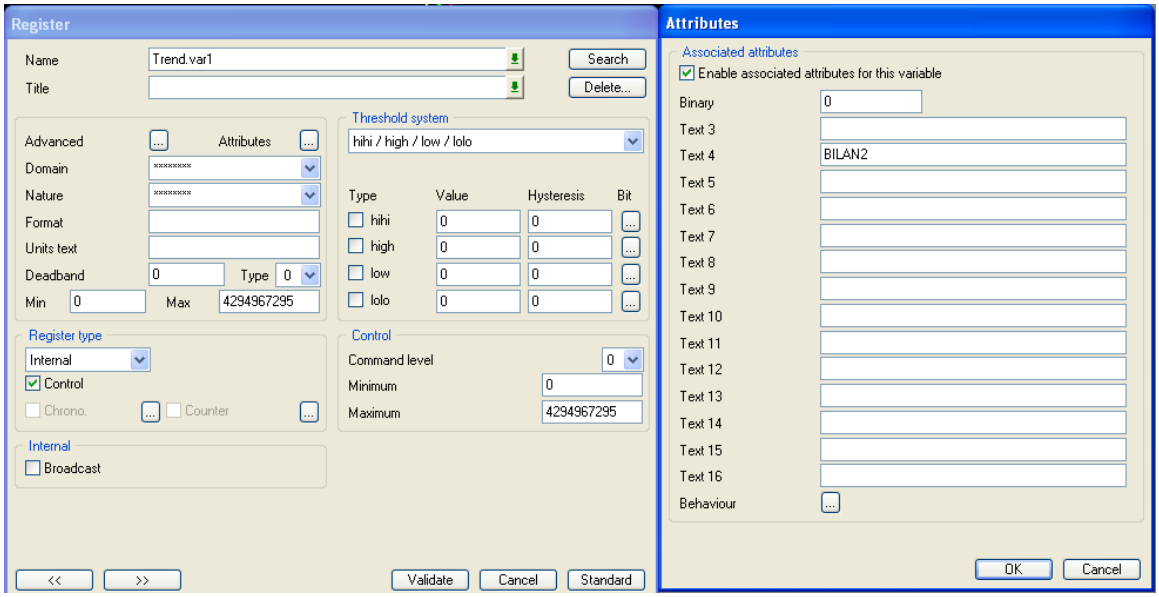


Figure 5: example attribute 4 setup a BILAN2

Remember don't put attributes a variables with suffix, only to variables that receive data.

This example has reference to HistoBlankMgr.ini file like that:

```
[HIS_PERIODTREND1]
AttributeNumber         = 4
AttributeValue          = BILAN2
TriggerVariableName     = His.TrendTrigger
TimeStamped             = 12:02:00
Period                  = 60
Synchronize             = 1
Suffix                  =
Dateshift               = 0
NotifyTriggerVariableName = His.NotifyTrigger
```

Visual C++ project

The main code is in svmgrHisTrendPeriod.cpp file.
The main description is in svmgrHisTrendPeriod.h file
The file svmgrHisTrendPeriodIntf.cpp contain implemented methods of ISVMgr interface.

The objective of this user DLL is updating the time stamp of variables defined with an attribute number and attribute value specified in .ini File.

There is a trigger variable (TriggerVariableName), if it change its value, will force the writing of TimeStamped over the variables defined with attribute value specific but with the suffix defined (if suffix doesn't exist, will be executed over the same variable).

There is a Period defined in .ini File. This define the period that the DLL will rewrite over the variables with suffix (if suffix doesn't exist, will be executed over the same variable) the actual time with the actual value. The synchronize field is to synchronize the beginning of Period.

Explaining generally for each class:

struct ISVMgrSpecific : public ISVMgr
Implementation of user dll interface.

class _Var
Used for represent a variable.

class _AdvisedAnaVar: public _Var
Used for subscribe to a register variable.

Explaining generally for each function:

StartProject: This function is global. Read the .ini File, set the init values over the variables and make call to **StartAdvise**, this last function Advise the variable trigger and all the variables with the attribute value defined in .ini File.

StopAdvised: This function un-advise all the variables advised at the Start.

OnDataChange: This function will be called when change the value of any variable advised, but if change the "trigger variable" will call UpdateVariablesAdvised with the

time to setup (Only if the before state of “trigger variable” is valid, that is to say different of NS)

SetValue: Is used to save the last value and status. Afterwards will be asked to compare with condition NS.

UpdateVariablesAdvised: Will write over the variable with suffix (if suffix doesn’t exist, will be executed over the same variable) the time defined in function inputs. Only work with analog variables.

OnTimerElapsed: In the event of elapsed time, this function will make the first step to update the entire variables with suffix (if suffix doesn’t exist, will be executed over the same variable) with the values of origin variables but with the actual time. The first time wait for the synchronization, after that begin the cyclic call to timers with the Period defined in .ini File. Each time will call UpdateVariablesAdvised to update the variables.

Installing process

The steps necessary to install are:

If C:\SV is the installation folder of Scada sofware

1- Copy the project in ScadaProject folder into the folder C:\SV \usr\
2- If you want, you can rename ScadaProject folder to svmgrHisTrendPeriod folder
3- Into the C:\SV\bin folder, copy the UsrMgr.dat files.
4- After compilation, copy svmgrHisTrendPeriod.dll Into the C:\SV\bin folder (if you install visual c++ project in the C:\SV\ManagerToolkit\Source\ svmgrHisTrendPeriod \BlankManagerProject folder, the DLL will be generated automatically in the SV\bin folder).