



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Juha Loponen

CREATION OF ABB MICROSCADA DATABASE TOOL

School of Technology
2018

ABSTRACT

Author	Juha Loponen
Title	Creation of ABB MicroSCADA Database Tool
Year	2018
Language	English
Pages	31
Name of Supervisor	Jari Koski

The aim of this final thesis project was to create a new tool for creating MicroSCADA databases. The current means of database creation are not up to date with the workload of the projects and the user-friendliness standards.

There was a desire to get the tool as an independent program that does not need a MicroSCADA installation on the engineering-PC and to clarify the process of creating database objects in list based environment. The user-friendliness was a key aspect in creating the new tool and it was done by adding a vast checking system to the tool by listing default values of database objects and comparing user inputs to them. The tool was created in Microsoft Excel with the Visual Basic for Application –code.

The new tool is efficient and has a clear user interface. It is most useful in big projects with many similar signals in different stations and bays.

TABLE OF CONTENTS

TIIVISTELMÄ

ABSTRACT

INTRODUCTION	6
1 ABB.....	7
1.1 ABB Grid Automation and MicroSCADA.....	7
2 BACKGROUND AND THE NEED FOR A NEW TOOL	8
2.1 Standard Function	8
2.2 LBE tool.....	14
3 CREATING NEW DATABASE TOOL.....	17
3.1 Requirements of New Tool.....	17
3.2 Designing SCADA Interface and User Interface.....	17
3.3 Functionalities and validations	19
3.3.1 Validate	19
3.3.2 Clear Log.....	22
3.3.3 Select output path.....	23
3.3.4 Create Files.....	24
4 TESTING	25
4.1 Testing of Database Tool	25
4.2 Testing the Interface with MicroSCADA	27
4.3 Testing the Actual Use of Database Objects in MicroSCADA	27
5 CONCLUSIONS	30
REFERENCES.....	31

LIST OF FIGURES

Figure 1 Standard function dialog with list of standard function types	9
Figure 2 Example of standard function dialog with fewest options, Line Indicator	10
Figure 3 First half of the biggest standard function creation dialog with most options, Circuit breaker	11
Figure 4 Second half of the biggest standard function creation dialog with options, Circuit breaker.....	12
Figure 5 Accessing the Process Object Tool from Standard Function dialog	13
Figure 6 Creating indexes from Process Object Tool	14
Figure 7 Group properties view in MicroSCADA.....	16
Figure 8 Layout of database sheet.....	18
Figure 9 Controls of the main sheet and the log of program	18
Figure 10 Time sequence diagram of validate function.....	19
Figure 11 Errors that occurred when validation failed on OI settings	20
Figure 12 Example of attributes defined in DBDefaults sheet	21
Figure 13 Example of an error from inconsistent standard function group with differing protocols	22
Figure 14 Log of a successful validation	22
Figure 15 Time sequence diagram of clear log function.....	23
Figure 16 Time sequence diagram for selecting output path	23
Figure 17 Time sequence diagram of file creation.....	24
Figure 18 View of database in the program	28
Figure 19 Database imported to MicroSCADA.....	28
Figure 20 Graphic objects in MicroSCADA monitor pro and a dialog of an alarm indicator with simulated values.....	29

INTRODUCTION

The aim of this final thesis project is to design and create new database creation tool for MicroSCADA for ABB Grid Automation Systems project team.

Existing means of creating and modifying databases especially on larger projects are not efficient. Projects with many signals are common to have changes during the projecting period and can cause unnecessary extra work in projects.

Creating a working, efficient and user-friendly tool can help project engineers do less of inefficient but commonly occurring work with the modification of database in cases, such as changed naming policies or adding or changing of signals. With the ease of modifying and then creating a new correct database it is possible to save engineering hours needed to complete projects.

1 ABB

ABB was founded in 1988 as a result of a fusion between the electrical technologies of Swedish Asea and Swiss Brown Boveri. Today ABB is a pioneer of technology with a wide portfolio from robotics to grid solutions. ABB operates in more than 100 countries and employs around 136000 people. ABB headquarters are in Zurich, Switzerland. ABB is divided in four Business units, which are Electrification products, Robotics and motion, Industrial automation and Power grids. /1-3/

ABB has 5100 people working in Finland and operates in 20 cities and towns. The main factory areas are in Hamina, Helsinki, Vaasa and Porvoo. ABB is one of the biggest employers in technical field in Finland and the biggest in Helsinki area. The revenue is around 2.2 billion euros. /4/

1.1 ABB Grid Automation and MicroSCADA

The Grid Automation business unit is part of ABB Power Grids division. Globally Grid Automation business unit employs around 4000 people and 60 in Finland. Grid Automation unit sells and uses MicroSCADA in projects. /5-6/

MicroSCADA name comes from Supervisory Control And Data Acquisition. It is a system that uses graphical interfaces, computer and networked communication for system control and management. MicroSCADA Pro is designed for complete functionality for real-time monitoring and control of primary and secondary equipment in transmission and distribution substations. It allows you to easily and safely interact with protection and control IEDs (intelligent electronic device), as well as with the process via the operator's workplace. /7/

2 BACKGROUND AND THE NEED FOR A NEW TOOL

Currently there are two main ways to create database: Standard function tool in MicroSCADA and LBE-tool (List based engineering) based on Excel which uses OPC connection to MicroSCADA. Projects have been growing in terms of number of signals and data that is being handled by MicroSCADA. The technical development of IED's and other field devices make the existing means of database creation and modification lacklustre in performance. This development project aims to combine the best means of current options and remove the issues that make them not ideal in the projects of today. The new database tool is designed and aimed to help project engineers to create and modify big projects with many signals with ease.

2.1 Standard Function

The Standard Function tool is an internal tool of MicroSCADA that creates all the necessary indexes and thus automatically fills all the database attributes the MicroSCADA needs to work properly. The user can choose his desired standard function type and the defining selections through dialogs, which can be accessed through the Object navigator in MicroSCADA tools. The devices that the user can create with the Standard Function tool are:

- Station
- Bay
- Switching Device
- Tap Changer
- Measurement
- Alarm Indicator
- Auto Reclose
- Trip Signal
- Generator
- Line Indicator

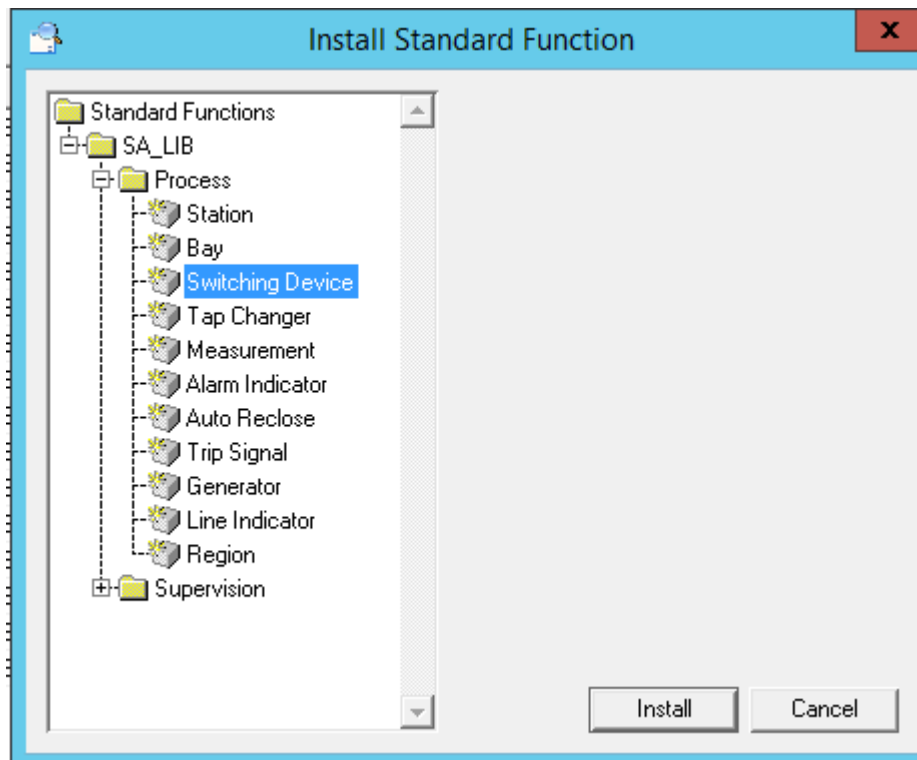


Figure 1. Standard function dialog with list of standard function types

The next step is to fill in the identifying attributes of database object. The fields that the user needs to fill in and the selection boxes and dropdown lists vary between the type of standard function that is chosen, but Logical name (LN) is an attribute that must be configured in all cases. LN is the main identifier of database object and it can include many indexes that have their own predetermined tasks. The object identifier is also reoccurring attribute which can be always filled in but it is not necessary from the perspective of MicroSCADA.

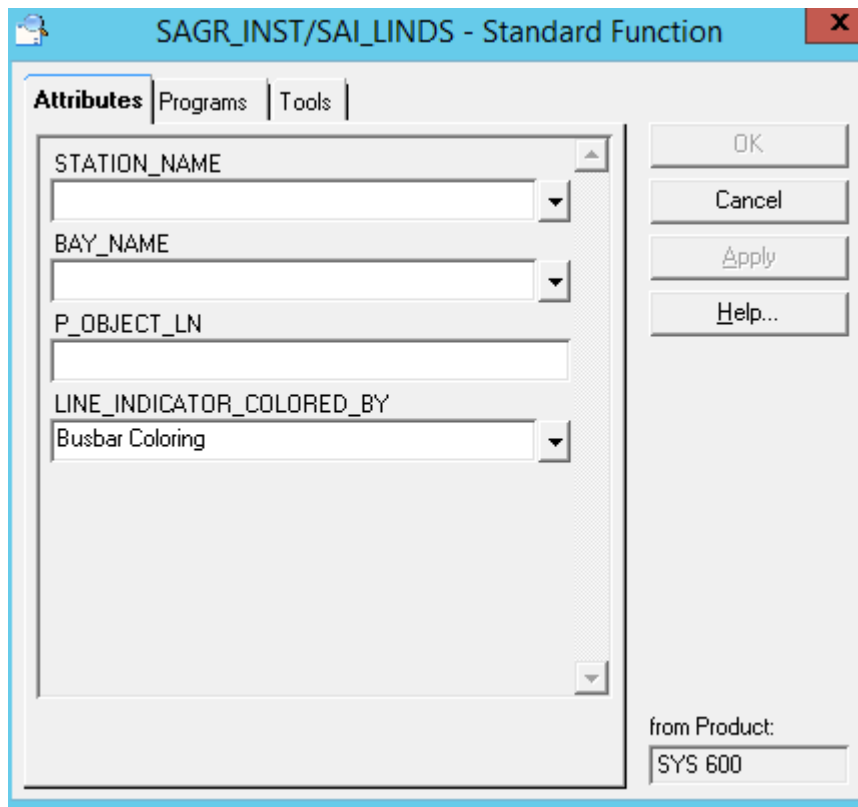
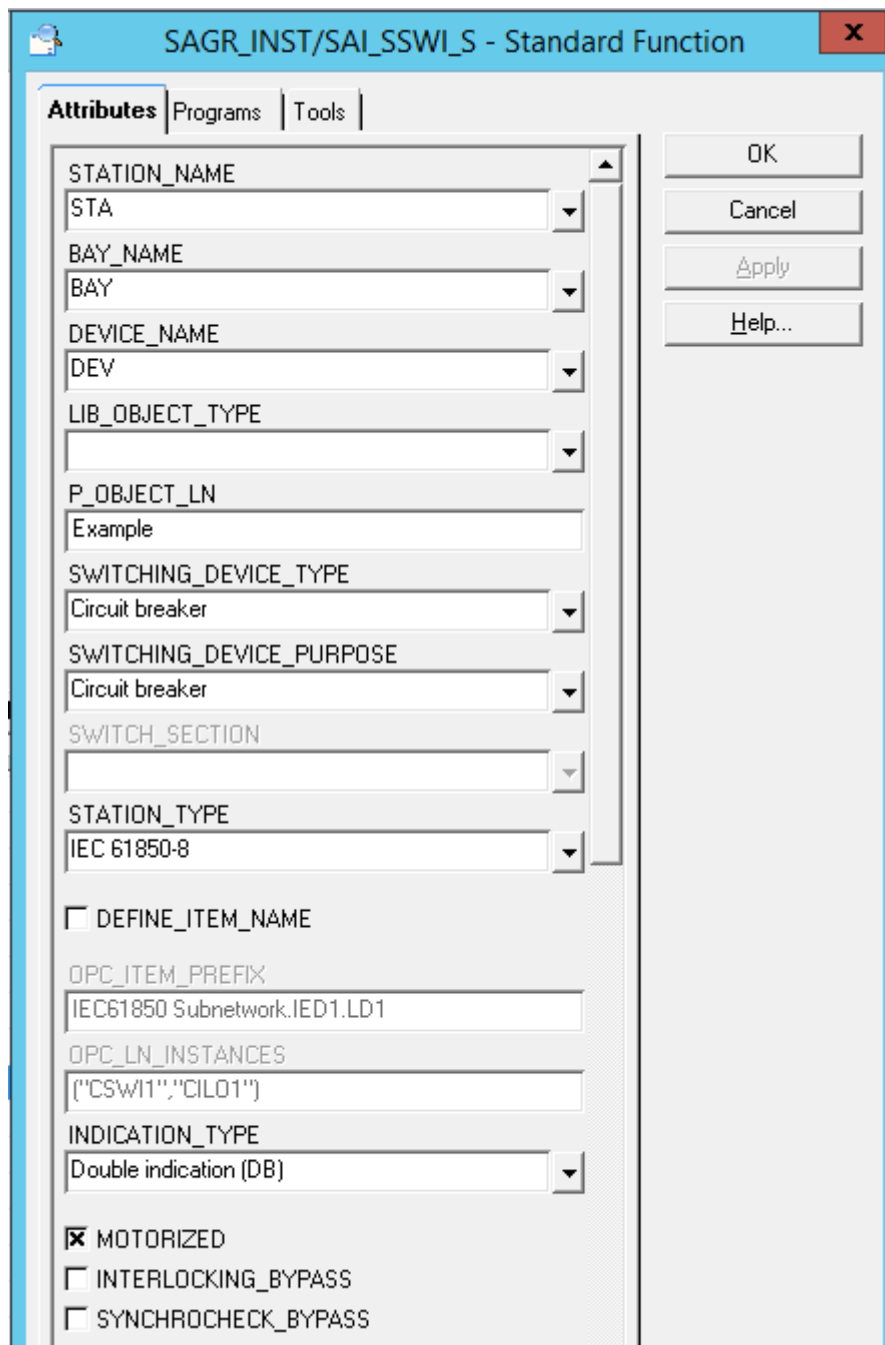


Figure 2. Example of standard function dialog with fewest options, Line Indicator

Another important attribute that is required to insert with most of standard function types is the Station_type attribute where the user chooses the protocol that is being used in data transmission from IED's and RTU's to MicroSCADA. The selection of specific Standard function type automatically changes Standard function dialog in some cases to match the needed attributes of chosen protocol to be available for selection or disables some options. There is a difference between needed user given attributes between protocol types and MicroSCADA fills in other attributes with default values.



SAGR_INST/SAI_SSWI_S - Standard Function

Attributes | Programs | Tools

STATION_NAME
STA

BAY_NAME
BAY

DEVICE_NAME
DEV

LIB_OBJECT_TYPE

P_OBJECT_LN
Example

SWITCHING_DEVICE_TYPE
Circuit breaker

SWITCHING_DEVICE_PURPOSE
Circuit breaker

SWITCH_SECTION

STATION_TYPE
IEC 61850-8

DEFINE_ITEM_NAME

OPC_ITEM_PREFIX
IEC61850 Subnetwork.IED1.LD1

OPC_LN_INSTANCES
["CSW1","CLO1"]

INDICATION_TYPE
Double indication (DB)

MOTORIZED

INTERLOCKING_BYPASS

SYNCHROCHECK_BYPASS

OK
Cancel
Apply
Help...

Figure 3. First half of the biggest standard function creation dialog with most options, Circuit breaker

The screenshot shows a configuration dialog box with the following fields and options:

- CONTROL_TYPE**: Secured cmd with 5 AOs
- CMD_PARAMETER**: Select-Operate
- OUTPUT_STATUS**
- CONTROL_PULSE_LENGTH**: 0
- CONTROL_BITS**: 0
- QUALIFIERS**: 0
- AUTHORIZATION_GROUP**: MV_CONTROL
- OPERATOR_PLACE_HANDLING**: None
- BAY_LR_POLARITY**: Remote=0,Local=1
- TAGOUT**
- AUXILIARY_PLUG**
- EVENT_RECORDING**
- ADD_CAUSE_IX**: 55
- from Product:** SYS 600

Figure 4. Second half of the biggest standard function creation dialog with options, Circuit breaker

When the user has finished filling in the attribute tab of the dialog, they can use the tool tab where the user can access the process object tool to create the previously configured database objects. This creates all the necessary indexes under the defined Logical name. This process is reliable and fairly straightforward. One can be sure that there cannot be errors in these database objects regarding the functionality of MicroSCADA, although user errors may still occur. The issue with this creation style is that by following the previous procedure, only one logical name is created, but projects may have thousands of logical names and the creation in this way takes too much time to be efficient.

The tool does not fill in the addresses of the objects so those have to be filled in one by one after creation. Filling in the addresses is necessary part of configuring

database and it will occur in every project. The modification of existing Logical names is not possible in MicroSCADA either and the user would be forced to create all new points if the naming policy changes in any way.

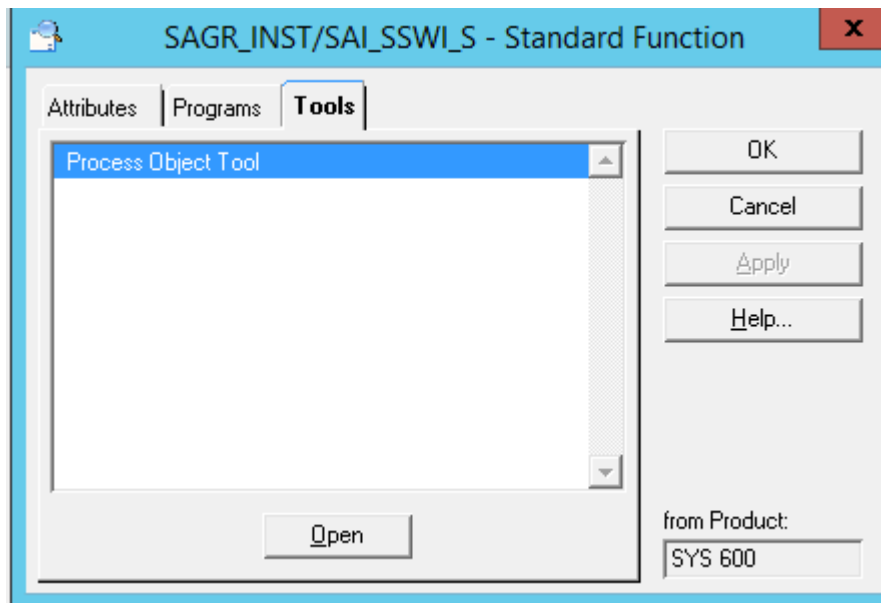


Figure 5. Accessing the Process Object Tool from Standard Function dialog

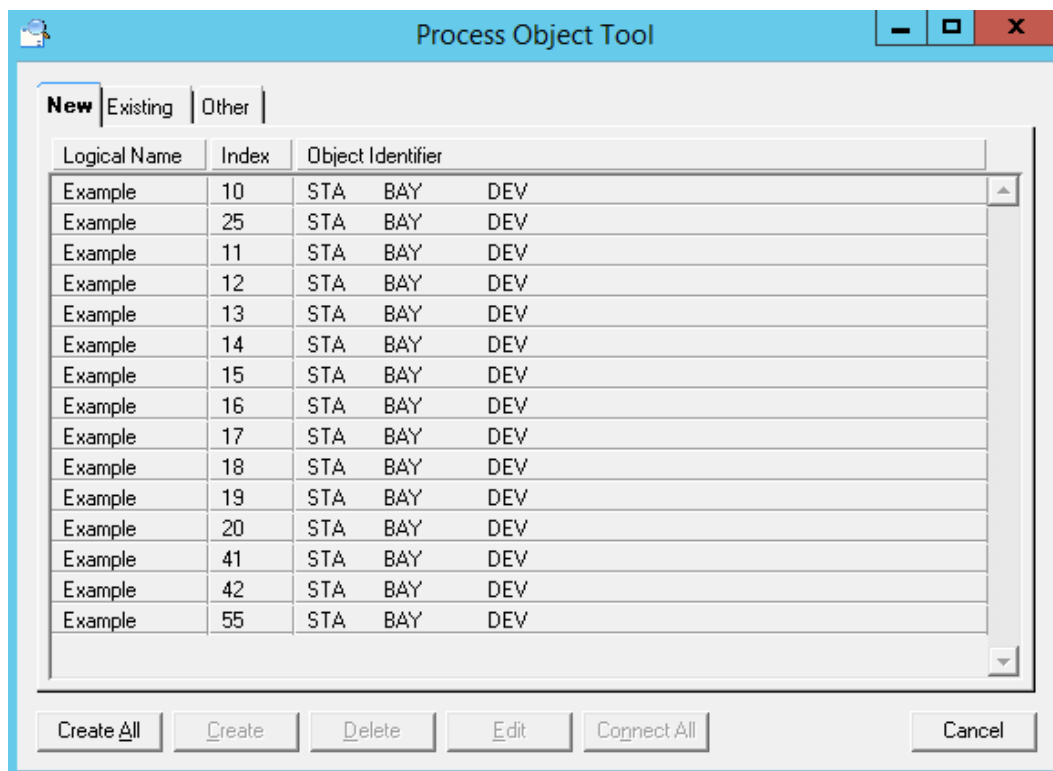


Figure 6. Creating indexes from Process Object Tool

2.2 LBE tool

List based engineering-tool is an engineering tool based on Excel and it comes with the installation of MicroSCADA. It is an earlier creation that is designed solely for project engineering. It is an external function out of MicroSCADA and it communicates through the OPC server. LBE functions by running a SCIL code to SCADA through the OPC server. Even though the user needs to create and modify the database objects in Excel, they will be created straight to the SCADA database when they are created by LBE and there is no need for manual importing of data. This type of interface definitely has benefits but there are drawbacks as well. It gives the user ease of use because of the straight forward implementing of database objects to MicroSCADA but requires that the user is either working on the computer that has the MicroSCADA installed or the user has a virtual machine which is a replicate of the computer. In the latter case, the direct creation of MicroSCADA database is lost but in this case the user can export the database from virtual machine and import

it to the server. When using this method on servers, it requires that Excel is installed on the server and that adds an unnecessary cost to projects.

LBE is better option than the Standard function tool to create mass of database objects as it makes it easier to copy and multiply database objects. When the user has completed one row which means creating one logical name with all indexes, the user can copy and paste the row and just by changing LN he can multiply signals. This is very useful in bigger projects with many signals of the same type. One of the disadvantages in the LBE layout is that all of the indexes of LN are on the same row and the modification of indexes can be disorientating and the user mistakes are easy to make when adding new indexes. This means that the user must add all the needed attributes of the indexes to the same row. It is easy for the user to make corrections to a wrong row because the logical name and the attributes the user wants to correct cannot be visible at the same time in most of the cases due to the way the layout and indexes are modified in LBE.

Using LBE is not as simple as using the standard function tool. Prior to creating any points in LBE the user must create examples of the standard function types he wants to use. When viewing the group properties of the EXAMPLE database objects created before the attributes that are necessary to fill in the LBE can be seen.

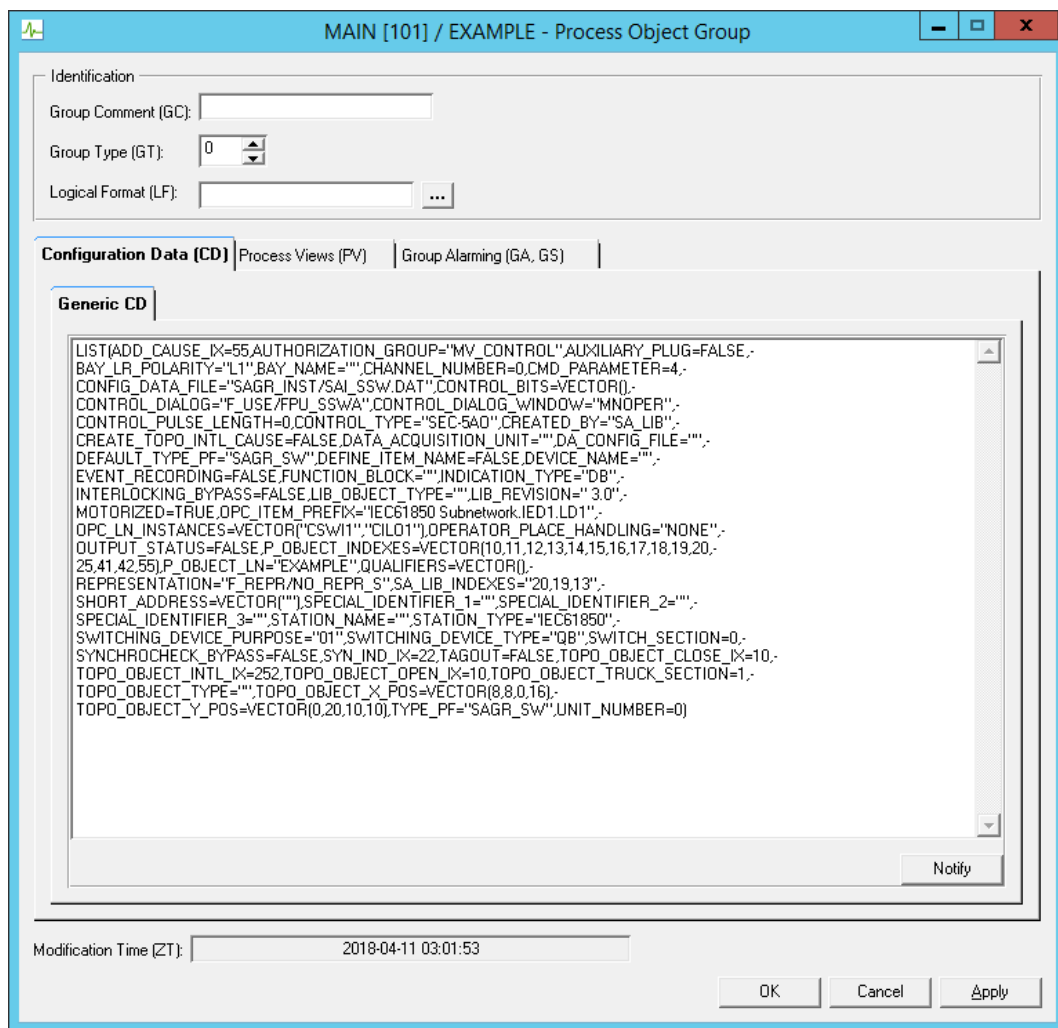


Figure 7. Group properties view in MicroSCADA

The LBE tool has received some upgrades that have been in unofficial distribution with project engineers but none of the versions have not been perfect so there is a need for a completely new tool.

3 CREATING NEW DATABASE TOOL

3.1 Requirements of New Tool

The goal for the new tool is that the functionalities it possess should outperform the existing means of database creation. The list below shows the aspects that were thought of when thinking what the new tool should ideally be able to fulfil.

- More streamlined list based modification
- More explanatory information to help the user fill in values
- MicroSCADA installation on engineering PC is not mandatory
- Comprehensive user input validation to mitigate errors
- Error handling and help on solving errors
- Exports database from MicroSCADA in to the tool
- Use existing interface for data transfer between MicroSCADA and tool

3.2 Designing SCADA Interface and User Interface

The planning of the new tool started from selecting the interface, how the database is moved from our tool to the MicroSCADA. Going through the options of importing and exporting databases of MicroSCADA, the CSV (Comma separated value) import through Import/export tool was selected as the most efficient mean. The CSV files are easy to create with Excel and the MicroSCADA support for CSV import is not expected to be removed from MicroSCADA tools in future updates. Import/export tool will be kept up to date in future updates.

When using CSV files, the program will create more than one Excel file, the header file which tells SCADA what type of other CSV files it has to read. Types that can be included are all of the different signal types: AI, AO, BI, BO, BS, DB, DI, DO, PC and a so-called P-file, which has the standard function specific CD attributes. All of the different signal types are included in their own file. If some type of signal is not used, the file for that type will not be created. /8/

The next step was designing the user interface and layout with the desire to create a familiar looking interface. The aim was to create a layout where it is clear to the

engineer which cells needs to be filled in when the user is working on the database sheet and that the columns have explanation what the given attribute means and what kind of input user needs to fill in. The layout of the database sheet is made similar to the look of RTU Excel tool that is being used in projects because the homogenization of project tools was one of the key aspects.

A	B	C	D	E	F	G	H	I	J	K	L
Enabled	Standard function	Signal type	Protocol	Logical name	Index	Station	Bay	Device			Object text
Y/N				Text	Integer 1..65535	Text	Text	Text	Basic attributes		Text 63 characters
#ENBL	#STFU	#SIGT	#PCOL	LN	IX	OI.STA	OI.BAY	OI.DEV			OX
Y	Measurement	Analog Input - AI	IEC(101/104)	TEST_FD1	10 TEST	FD1					Breaker position indication
Y	Measurement	Binary Output - BO	IEC(101/104)	TEST_FD1	13 TEST	FD1					Breaker open execute command
Y	Measurement	Binary Input - BI	IEC(101/104)	TEST_FD1	15 TEST	FD1					Breaker device control block
Y	Measurement	Binary Input - BI	IEC(101/104)	TEST_FD1	16 TEST	FD1					Breaker open interlocked
Y	Measurement	Binary Input - BI	IEC(101/104)	TEST_FD1	17 TEST	FD1					Breaker close interlocked
Y	Measurement	Analog Input - AI	IEC(101/104)	TEST_FD1	18 TEST	FD1					Cause of interlocking
Y	Measurement	Analog Input - AI	IEC(101/104)	TEST_FD1	19 TEST	FD1					Breaker selection on monitor
Y	Measurement	Binary Input - BI	IEC(101/104)	TEST_FD1	20 TEST	FD1					Breaker command event
Y	Measurement	Cmd Termination - IEC	IEC(101/104)	TEST_FD1	113 TEST	FD1					
_END											

Figure 8. Layout of database sheet

The main control sheet of the program is called Main sheet. The database sheet, where the signals are modified and created, are on different tab than the controls of the program so it becomes clearer what the user is doing. The control buttons of the program are validating database, choosing output folder, clearing log, and creating signals. There is also a table for defining OI attributes. Read files button is for a function that is not implemented in this project.

Create Files

Read Files

Clear Log

Validate

#OI.NAME	#OI.LENGTH	#OI.TITLE	#OI.STD_NAME
OI.STA	10	Station	STATION_NAME
OI.BAY	15	Bay	BAY_NAME
OI.DEV	15	Device	DEVICE_NAME

Output Folder Path

... D:\

Output File Name

Application Name

MAIN

Timestamp	Type	Link	Log Text

Figure 9. Controls of the main sheet and the log of program

3.3 Functionalities and validations

All of the functionalities of the program have been done in the VBA (Visual Basic for Applications). Basic functionalities, such as clearing error log and choosing output folder are solely code but validating and creating the CSV files rely also on data tables filled with the default values of MicroSCADA attributes and examples of Standard functions.

3.3.1 Validate

Validate is the most important function in the program from the perspective of making sure that the database is filled with valid values and in a correct form. Validating database makes sure that there are default definitions for all the attributes that are used in creating indexes and when creating standard functions the groups are formatted accordingly.

Clicking the validate button runs the validate function which starts by checking that OI settings are correct with no duplicate definitions. Incorrect defining causes validation to stop and print errors to log on every validation step.

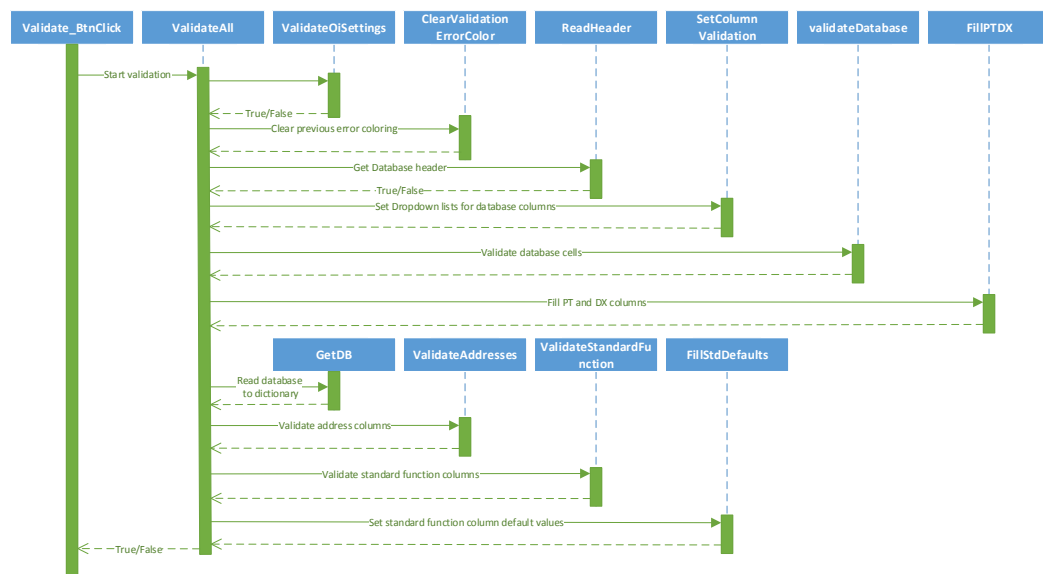


Figure 10. Time sequence diagram of validate function

Create Files	Validate			#OI.NAME	#OI.LENGTH	#OI.TITLE	#OI.STD_NAME
				OI.REG	20	Region	STATION_NAME
Read Files				OI.REG	15	Station	STATION_NAME
				OI.BAY	15	Bay	BAY_NAME
Clear Log				OI.DEV	5	Device	DEVICE_NAME
				OI.VOL	5	Voltage	SPECIAL_IDENTIFIER_2
Timestamp	Type	Link	Log Text				
24.4.2018 13:26	Info		Validation started				
24.4.2018 13:26	Error		Duplicate OI Name in OI settings table on sheet [Main]				
24.4.2018 13:26	Error		Duplicate Standard function OI Name in OI settings table on sheet [Main]				
24.4.2018 13:26	Error		OI settings contain errors. Stopping execution				

Figure 11. Errors that occurred when validation failed on OI settings

In the next step the function clears error colouring created by previous validations. ReadHeader checks that all of the header rows attributes have a definition on the DBDefaults sheet. SetColumnValidation sets validation to columns that have dropdown lists that help the user fill the cells more easily. ValidateDatabase reads through the database and checks that the cells are filled with values that comply with definitions in the DBDefaults page. FillPTDX inserts values to PT and DX columns by cross-referencing user inputs to Protocol cell and Signal type cell on database. This mitigates errors because the user can choose correct values from clear text dropdown lists and the program inserts the code values to correct cells. GetDB builds the database rows and checks that there is no duplicate indexes on the same logical names and validates that the row is valid as a whole.

Attribute Name	Min Value	Max Value	Default Value	Allowed blank	Datatype	Only used in signal types
#DB_NAM	#DB_MIN	#DB_MAX	#DB_DEF	#DB_BLN	#DB_DTY	#DB_UNIQ
#ENBL	0	1		FALSE	text	
#STFU	0	64		TRUE	text	
#SIGT	0	64		FALSE	text	
#PCOL	0	64		FALSE	text	
AA	0	15	0	TRUE	number	
AB	0	1	0	TRUE	number	
AC	0	7	0	TRUE	number	
AD	0	65535	0	TRUE	number	
AE	0	1	0	TRUE	number	
AF	0	1	0	TRUE	number	
AG	0	5	1	TRUE	number	BI
AH	0	1	0	TRUE	number	
AN	0	63		TRUE	Text	
CE	0	1	0	TRUE	number	BI;BO;DB
CL	0	2147483647	0	TRUE	number	BI;BO;DB
CX	0	255		TRUE	text	
DP	-1	10	0	TRUE	number	AI;AO
DX	0	10		TRUE	text	
EE	0	1	0	TRUE	number	
EH	0	64		TRUE	text	
FI	0	2147483647	0	TRUE	number	

Figure 12. Example of attributes defined in DBDefaults sheet

ValidateAddresses goes through addresses in columns UN (Unit number), OA (Object address), OB (Object bits) and IN (Item name) and ON (OPC item name). These are the objects that hold the addresses that MicroSCADA uses in communication. ValidateStandardFunction checks if the user has picked a standard function type and that the type is valid. The function also checks that all of the indexes of one standard function group have the same UN and protocol. After this, the function checks the consistency of CD (Configuration data) attribute. This attribute is special to the standard function and it has overlapping values with the MicroSCADA default attributes. The function finds the fields that would have overlapping values, inserts the values there automatically and thus reduces unnecessary work. The last checking is in the situation where the user has selected measurement as a standard

function type. The measurement group can have only 1-4 signals in one group. The function checks that the limit is not exceeded.

A	B	C	D	E	F	G	H	I	J	K	L
Enabled	Standard Function	Signal Type	Protocol	Logical name	Index	Region	Station	Bay	Device	Voltage	Object text
Y/N				Text	Integer 1..65535	Text	Text	Text	Text	Text	Text 63 characters
#ENBL	#STFU	#SIGT	#PCOL	LN	IX	OI.REG	OI.STA	OI.BAY	OI.DEV	OI.VOL	OX
Y	Alarm Indicator	Binary Input - BI	Modbus	*jhelmaalarm61850	10	ohjelma	alarm	61850			Alarm1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	11	ohjelma	alarm	61850			Alarm2
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	12	ohjelma	alarm	61850			Alarm3
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	13	ohjelma	alarm	61850			Alarm4
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	14	ohjelma	alarm	61850			Alarm5
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	15	ohjelma	alarm	61850			Alarm6
Y	Alarm Indicator	Double Binary - DB	IEC 61850	Ohjelmaalarm61850	16	ohjelma	alarm	61850			Alarm7
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	17	ohjelma	alarm	61850			Alarm8
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	18	ohjelma	alarm	61850			Alarm indicator blockings
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	19	ohjelma	alarm	61850			Alarm indicator selected on monitor

Figure 13. Example of an error from inconsistent standard function group with differing protocols

When all of these validations have run, ValidateAll completes and writes RX (Reserved text) attribute to standard function indexes. After this, the validation is complete and the user gets a validation completion message.

Timestamp	Type	Link	Log Text
24.4.2018 15:54	Info		Validation started
24.4.2018 15:54	Info		Validating database...
24.4.2018 15:54	Info		Validating standard functions...
24.4.2018 15:54	Info		Validation complete

Figure 14. Log of a successful validation

3.3.2 Clear Log

The clear log function, as name implies, clears the tool log and is run when the user clicks the Clear log –button. The function figures out the area between the main sheet header and the first empty cell in log area. Then creating a range between these cells the function gets the range where there are log references and request a clearing of the cell values.

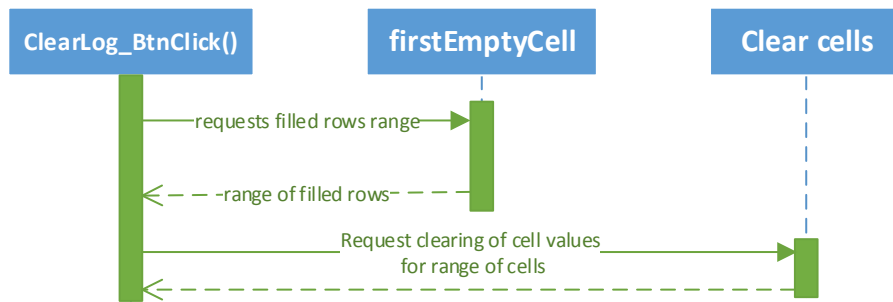


Figure 15. Time sequence diagram of clear log function

3.3.3 Select output path

Select output path is used to select the folder where the user wants database files to be created in. On button click the function checks the output folder path cell and opens the directory in Windows Explorer. The user can choose a desired path from the Explorer or write it directly to the cell in the program.

If the value is not for an existing path or the path is otherwise not correct the current path of database tool is opened and set as a path.

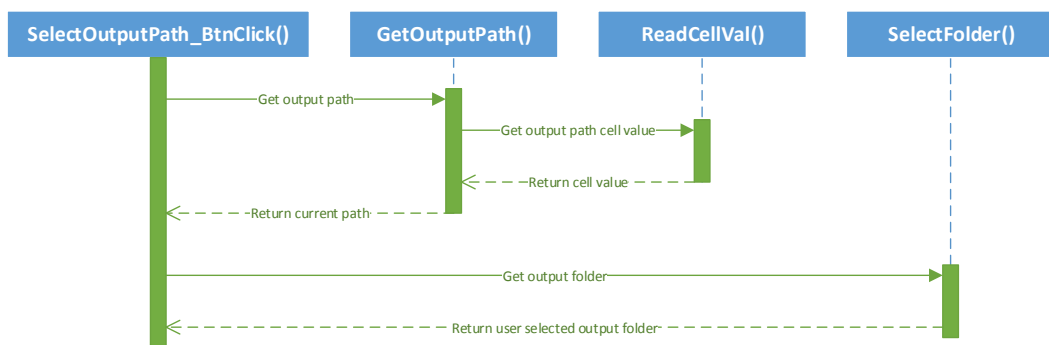


Figure 16. Time sequence diagram for selecting output path

3.3.4 Create Files

Create files is the most complicated function in the program, it has multiple steps of its own but also runs the previously explained ValidateAll function. The function starts by running the validations. After this, the program starts preparing the writing of the files. The program needs to split the database by the signal types, then it can continue and create the divided files one at a time. The next step is to check if there is any standard function rows in the database, and if there is then create the P-file. The last step is to write the header file which includes all of the file names that are included in the output. Import/Export tool of MicroSCADA reads this file.

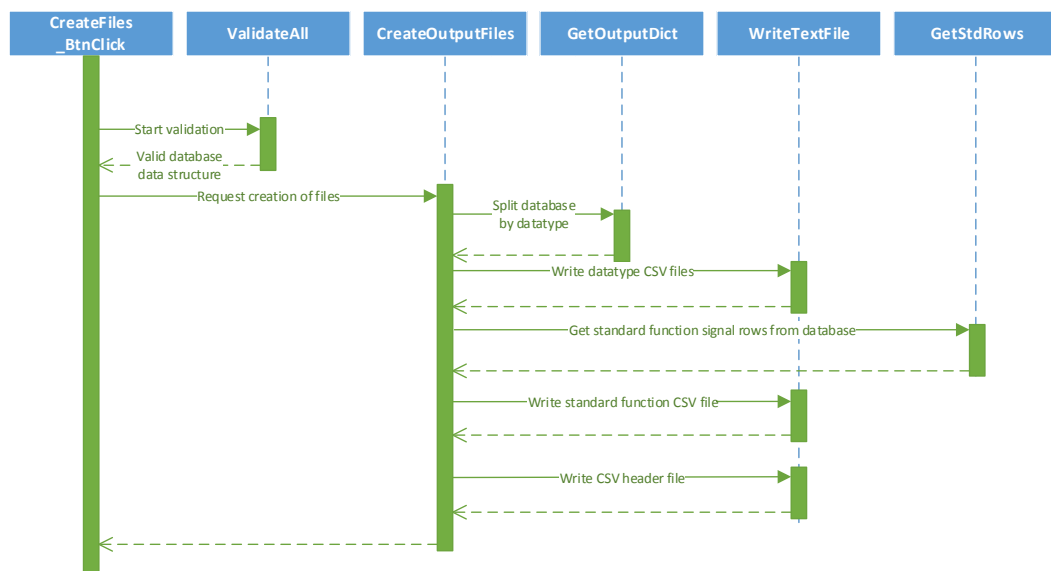


Figure 17. Time sequence diagram of file creation

4 TESTING

4.1 Testing of Database Tool

The testing of the tool was mostly done by trying to get the checking functions to work with typing in incorrect values to fields, which have validation. There are also fields that have no validation so checking how inputting invalid values in these fields affect the tool was also a thing to consider. Duplicate value checking is also in a major role in the tool so checking that it functions correctly is essential.

The Main sheet has an OI attribute definition table that has free text fields. The settings of the attributes are done in MicroSCADA SYS_BASCON and the settings in tool has to follow those. The tool cannot check if the user has typed the correct definition or not. The tool checks that OI.NAME, OI.TITLE and OI.STD fields do not have duplicate values and that the input is not over 64 characters. The OI.LENGHT field has to be numbers or an error occurs. The user can leave fields empty without getting errors, but this is how it is intended to work. All of the five definitions are not always used, three definitions indicating station, bay and device is a common setup.

Other user input fields on the Main sheet are Output Folder Path, Output File Name and Application Name. The application name is a free text cell with no validation. The user must type the MicroSCADA application name that is used in the project. This field is only for creating Station objects. Station objects need to have an application name when creating them or they will not work correctly. Output File Name and Output Folder Path follow the same naming principles as all windows files. Illegal characters are:

- < (less than)
- > (greater than)
- : (colon)
- " (double quote)
- / (forward slash)
- \ (backslash)

- | (vertical bar or pipe)
- ? (question mark)
- * (asterisk)

The database sheet has a lot more cells that the user interacts with and many cells with validation. All of the validation that checks if the type of input is in correct form, either text or number, is done by the same logic so if one cell of a certain type works correctly, then all others should also. Additionally, the checking of each MicroSCADA attribute was done with the intent of checking that the values that have been inserted to DBDefaults as limits are correct. This was done by cross-checking DBDefaults table with MicroSCADA Application objects manual and making sure that the code handles the limits and default values correctly.

There are two types of text cells, a number cell type, a Boolean cell type and a vector cell type. Text can be in a logical name format where it has restrictions with special characters, use of Nordic characters and space and the character limit. Another type is the free text format where it only has character limitation. Errors on number cell come from typing in anything other than integers and if the number is bigger or smaller than the limits allow. Boolean needs to be one or zero. Vector format expects list of comma separated values.

Some of the more important fields that are not easy to see if they are incorrect were made so that user cannot accidentally make mistakes that are hard to solve. These fields are filled by the tool based on the other inputs of the user.

The database sheet was also tested with many foreseeable user mistakes, a few of important ones are listed below. All of these cases are handled and the log message gives information accordingly.

- Typing the same LN IX combination twice
- Inserting indexes that are not defined for the standard function type
- Adding a column to sheet header
- Creating empty rows in between signals

Creating and validating the database as a whole was the point where errors were expected and they were present. Most of the errors that occurred were caused by missing some information when filling in the default tables. The validations of the tool worked well and the log made it easy to see where the problems were and the correcting measures were easy to make. Fields with no validation because it is impossible to make without connection to MicroSCADA are an issue that is left to be handled by counting on that the user is a professional and has the help of the manual and realizes that it must be checked manually.

4.2 Testing the Interface with MicroSCADA

Testing the interface and the import function of MicroSCADA with the files that are created with Database Tool was simply done by creating files and trying to import them. The fact that the program is creating files properly does not mean that the files are completely in a correct format. Some minor issues were present with the standard function P-file. When importing the files MicroSCADA reads the P-file as SCIL and the import tool gives errors if there is a slightest error in the syntax. Most of the errors came from incorrectly filled attribute values in default value tables.

4.3 Testing the Actual Use of Database Objects in MicroSCADA

The last step is to check if the Display builder graphic objects work correctly and dialogs open normally. All of the possible Standard functions that the Database tool can create were made and the database was imported to a virtual machine which has a MicroSCADA 9.4 installation. The Standard functions that can be created with the Database tool were cut from original 10 to 7. These were shortlisted out of the group as useful functions that are still used in new projects. The Standard functions that database tool can create:

- Alarm indication
- Bay
- Line indicator
- Measurement

- Station
- Switching device
- Tap changer

The functionality of the objects were tested by dragging the object from the object browser in MicroSCADA display builder and then opening the dialogs. If there are errors in the CD attribute then the dialogs will not open. By getting the dialogs to work the proof of concept can be given for this database tool. The dialogs worked without any big issues.

Y/N				Text	Integer 1..65535	Text	Text	Text	Text	Text	Text 63 characters	Boolean
#ENBL	#STFU	#SIGT	#PCOL	LN	IX	OI.REG	OI.STA	OI.BAY	OI.DEV	OI.VOL	OX	IU
Y	Switching device	Double Binary - DB	IEC 61850	Earthswitch	10	ohjelma	SW		61850		Breaker position indication	1
Y	Switching device	Analog Output - AO	IEC 61850	Earthswitch	11	ohjelma	SW		61850		Breaker open execute command	1
Y	Switching device	Analog Output - AO	IEC 61850	Earthswitch	12	ohjelma	SW		61850		Breaker close execute command	1
Y	Switching device	Analog Output - AO	IEC 61850	Earthswitch	13	ohjelma	SW		61850		Breaker open execute command	1
Y	Switching device	Analog Output - AO	IEC 61850	Earthswitch	14	ohjelma	SW		61850		Breaker close execute command	1
Y	Switching device	Analog Input - AI	IEC 61850	Earthswitch	15	ohjelma	SW		61850		Breaker device control block	1
Y	Switching device	Binary Input - BI	IEC 61850	Earthswitch	16	ohjelma	SW		61850		Breaker open interlocked	1
Y	Switching device	Binary Input - BI	IEC 61850	Earthswitch	17	ohjelma	SW		61850		Breaker close interlocked	1
Y	Switching device	Analog Input - AI	IEC 61850	Earthswitch	18	ohjelma	SW		61850		Cause of interlocking	1
Y	Switching device	Analog Input - AI	IEC 61850	Earthswitch	19	ohjelma	SW		61850		Breaker selection on monitor	1
Y	Switching device	Binary Input - BI	IEC 61850	Earthswitch	20	ohjelma	SW		61850		Breaker command event	1
Y	Switching device	Analog Output - AO	IEC 61850	Earthswitch	25	ohjelma	SW		61850		Breaker cancel command	1
Y	Switching device	Binary Input - BI	IEC 61850	Earthswitch	41	ohjelma	SW		61850		Breaker open blocked	1
Y	Switching device	Binary Input - BI	IEC 61850	Earthswitch	42	ohjelma	SW		61850		Breaker close blocked	1
Y	Switching device	Analog Input - AI	IEC 61850	Earthswitch	55	ohjelma	SW		61850		Add cause of command	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	10	ohjelma	alarm		61850		Alarm1	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	11	ohjelma	alarm		61850		Alarm2	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	12	ohjelma	alarm		61850		Alarm3	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	13	ohjelma	alarm		61850		Alarm4	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	14	ohjelma	alarm		61850		Alarm5	1
Y	Alarm Indicator	Binary Input - BI	IEC 61850	Ohjelmaalarm61850	15	ohjelma	alarm		61850		Alarm6	1
Y	Alarm Indicator	Double Binary - DB	IEC 61850	Ohjelmaalarm61850	16	ohjelma	alarm		61850		Alarm7	1
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	17	ohjelma	alarm		61850		Alarm8	1
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	18	ohjelma	alarm		61850		Alarm indicator blockings	1
Y	Alarm Indicator	Analog Input - AI	IEC 61850	Ohjelmaalarm61850	19	ohjelma	alarm		61850		Alarm indicator selected on monitor	1
Y	Line Indicator	Analog Input - AI	ANSI	OhjelmaLINE104	10	ohjelma	line	ANSI			Line indicator	1
Y	Line Indicator	Binary Input - BI	ANSI	OhjelmaLINE104	253	ohjelma	line	ANSI			Virtual switch for Topol. Col.	1
Y	Line Indicator	Analog Input - AI	ANSI	OhjelmaLINE104	254	ohjelma	line	ANSI			Ext. ground ind. for Topol. Col.	1
Y	Line Indicator	Analog Input - AI	ANSI	OhjelmaLINE104	255	ohjelma	line	ANSI			Infused color for Topol. Col.	1
Y	Line Indicator	Analog Input - AI	ANSI	OhjelmaLINE104	50000	ohjelma	line	ANSI			Voltage level	1
Y	Measurement	Analog Input - AI	IEC101/104	OhjelmaMeas104	10	ohjelma	meas		104		Current L1	1
Y	Measurement	Analog Input - AI	IEC101/104	OhjelmaMeas104	11	ohjelma	meas		104		Current L2	1
Y	Measurement	Analog Input - AI	IEC101/104	OhjelmaMeas104	12	ohjelma	meas		104		Current L3	1
Y	Measurement	Analog Input - AI	IEC101/104	OhjelmaMeas104	13	ohjelma	meas		104		Neutral current ID	1

Figure 18. View of database in the program

OHJELMAALARMI61850	ID	ohjelma	Alarm1	FFHALAIALA	E7S
OHJELMAALARMI61850	11	ohjelma	Alarm2	FFHALAIALA	E7S
OHJELMAALARMI61850	12	ohjelma	Alarm3	FFHALAIALA	E7S
OHJELMAALARMI61850	13	ohjelma	Alarm4	FFHALAIALA	E7S
OHJELMAALARMI61850	14	ohjelma	Alarm5	FFHALAIALA	E7S
OHJELMAALARMI61850	15	ohjelma	Alarm6	FFHALAIALA	E7S
OHJELMAALARMI61850	16	ohjelma	Alarm7	FFHALAIALA	E7D
OHJELMAALARMI61850	17	ohjelma	Alarm8	FFHALAIALA	E6
OHJELMAALARMI61850	18	ohjelma	Alarm indicator blockings	FFHALAVBLK	E6
OHJELMAALARMI61850	19	ohjelma	Alarm indicator selected on monitor	FFHALAVMEV	E6
OHJELMALINE104	10	ohjelma	Line indicator	BCHMTD-CFD	9
OHJELMALINE104	253	ohjelma	Virtual switch for Topol. Col.	BCHMTD-CFPI	3
OHJELMALINE104	254	ohjelma	Ext. ground ind. for Topol. Col.	BCHMTD-CDCD	9
OHJELMALINE104	255	ohjelma	Infused color for Topol. Col.	BCHMTD-CFCD	9
OHJELMALINE104	50000	ohjelma	Voltage level	BCHMTD-CVOL	9
OHJELMAEAS104	10	ohjelma	Current L1	FFPMEAMCUR	C9
OHJELMAEAS104	11	ohjelma	Current L2	FFPMEAMCUR	C9
OHJELMAEAS104	12	ohjelma	Current L3	FFPMEAMCUR	C9
OHJELMAEAS104	13	ohjelma	Neutral current ID	FFPMEAMNCU	C9
OHJELMATAPSPA	10	ohjelma	Tap position	FPTAVRIPDS	N6
OHJELMATAPSPA	11	ohjelma	Tap ch. manual / auto ind.	FPTAVRIMAN	N7D
OHJELMATAPSPA	12	ohjelma	single / parallel ind.	FPTAVRISGL	N7S
OHJELMATAPSPA	13	ohjelma	Master / slave ind.	FPTAVRIMST	N7S
OHJELMATAPSPA	14	ohjelma	Tap ch. manual cmd.	FPTAVRDMAN	N1
OHJELMATAPSPA	15	ohjelma	Tap ch. auto cmd.	FPTAVRDMAN	N1
OHJELMATAPSPA	16	ohjelma	Tap ch. raise cmd.	FPTAVRCPDS	N1
OHJELMATAPSPA	17	ohjelma	Tap ch. lower cmd.	FPTAVRCPDS	N1
OHJELMATAPSPA	20	ohjelma	Tap ch. ext. blocking	FPTAVRIBLK	N7S
OHJELMATAPSPA	24	ohjelma	Voltage	FPTAVRMVOL	N6
OHJELMATAPSPA	29	ohjelma	Tap changer operation counter	FPTAVRDMR	N6
OHJELMATAPSPA	20	ohjelma	Select another monitor	FPTAVRMEV	N6
OHJELMATAPSPA	50000	ohjelma	Primary Voltage Level	FPTAVRCSVOL	N6

Figure 19. Database imported to MicroSCADA

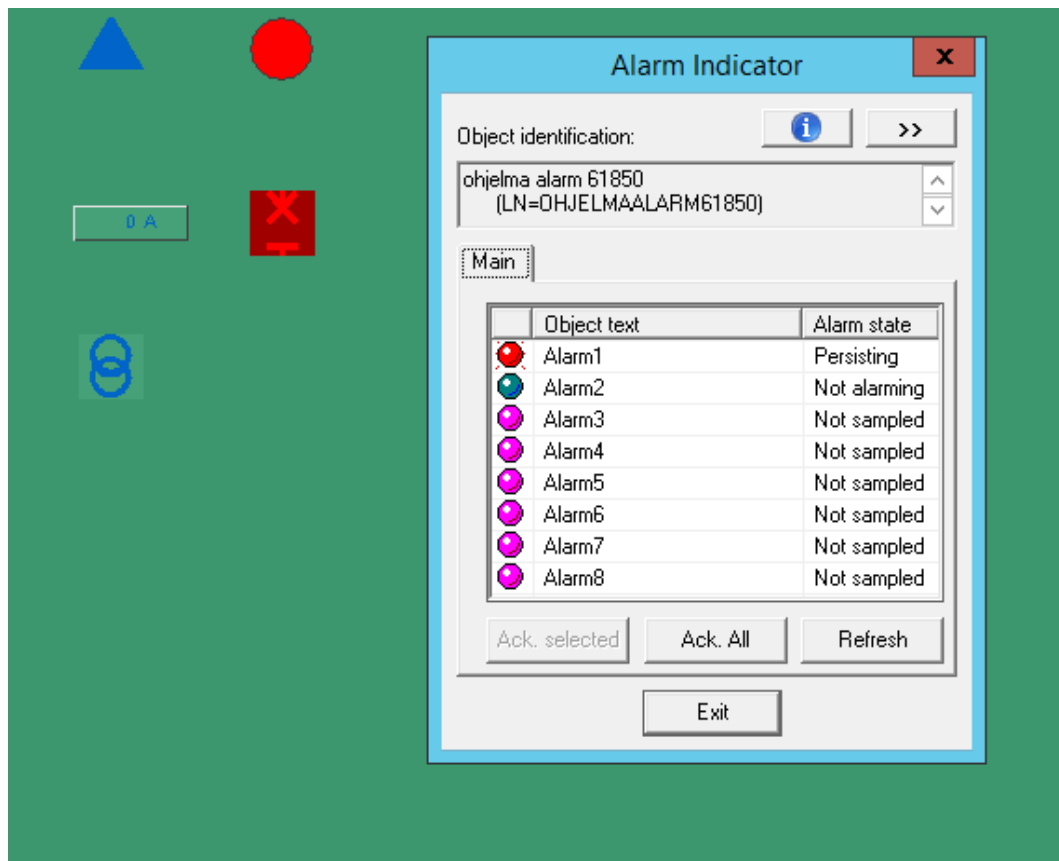


Figure 20. Graphic objects in MicroSCADA monitor pro and a dialog of an alarm indicator with simulated values

5 CONCLUSIONS

The Database tool meets most of the requirements set for the tool. The tool is stable, has easy to grasp functionalities and error solving that helps the user with information that is easy to understand. The tool can create standard functions as well as basic database objects and it has a wide variety of default values, which helps the user fill in cells more easily when creating standard functions. If there were a standard function index that would have been omitted, it is easy to add the definition to the default tables due to dynamic coding.

The tool is completely independent from MicroSCADA and it works with the existing import/export tool interface in SCADA.

The only thing that was not included was the reading of export files from MicroSCADA. This functionality had to be left out because of the amount of work would not have been possible to do in the given time. This would be an obvious point to continue in the development of this tool.

The tool with the current functionalities is still an improvement to the existing means when it was tested it in a small scale, although it will be tested properly when the pilot project will be done with the tool.

REFERENCES

/1/ ABB. Accessed 07.05.2018. <http://new.abb.com/about>

/2/ ABB lyhyesti. Accessed 07.05.2018. <http://new.abb.com/fi/abb-lyhyesti/historia>

/3/ ABB Businesses. Accessed 07.05.2018 <http://new.abb.com/about/our-businesses>

/4/ ABB suomessa. Accessed 07.05.2018 <http://new.abb.com/fi/abb-lyhyesti/suomessa>

/5/ Grid Automation Systems. Accessed 07.05.2018 <http://new.abb.com/fi/abb-lyhyesti/suomessa/yksikot/grid-automation-systems>

/6/ Grid Automation. Accessed 07.05.2018 <https://go.insideplus.abb.com/divisions/power-grids/bu-portals/grid-automation>

/7/ MicroSCADA Pro for Substation Automation. Product brochure ABB

/8/ MicroSCADA Pro SYS600 Application Design. Product manual ABB