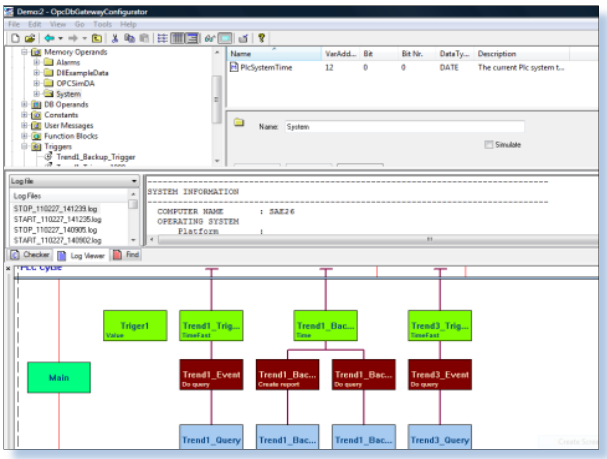
SAE – Automation – **Interoperability for your devices and applications.**

Integration of applications effectively

OpcDbGateway – configuring and programming, overview

Integration of applications with OpcDbGateway

OpcDbGateway enables integrating complex and easy software applications processing data from different connected external devices, databases and web services, to save them to databases, files or sending to the devices using different communication protocols.



Name	VarAdd...	Bit	Bit Nr.	DataTy...	Description
PlcSystemTime	12	0	0	DATE	The current Plc system t...

```
STOP_110227_141239.log
START_110227_141235.log
STOP_110227_140905.log
START_110227_140902.log
```

```
------------------------------------------------------------------
SYSTEM INFORMATION
------------------------------------------------------------------
  COMPUTER NAME        : SAE26
  OPERATING SYSTEM
     Platform          :
```

Figure 1 Configuration application

It can also start external programs with parameters configured by configuration application and also to procedures stored on databases.

Configuration application (Figure 1) of the OpcDbGateway enables to minimize time to integrate applications enabling to substitute laborious programming by configuring. The configuration is saved to the configuration database and used for controlling of the OpcDbGateway runtime application.

Multitude of the configurable components can be further enhanced according to the individual needs. There is a possibility to program own modules (enhancing Dll's) using languages as C++ or C#. If the defined API is respected, these modules are becoming a part of the OpcDbGateway configurable functionality. This way, the functionality can be enhanced without limitations. Functionality enhancements achieved by using of different script languages either proprietary or standard is used also in other SCADA systems, but it is necessary to learn them and, contrary to the standard programming languages, they not always provide satisfying performance.

Laborious programming is substituted by configuring. High flexibility – new user's programmed modules can enhance configurable functionality

OpcDbGateway uses preferably communication interface according to the OPC standard for the communication with external devices. OPC servers communicate with devices over proprietary protocol, and with the OPC client built in the OpcDbGateway over OPC interface. They are delivered mostly with own configuration application. To eliminate a duplicate configuring work, the configuration application of the OpcDbGateway enables **mapping of address space of the external OPC server** to own configuration.

OpcDbGateway can be used also for integrating of database enterprise applications. Database tables of such applications can be easily mapped to the OpcDbGateway using mapper in configurator. It is possible to work in opposite direction – to create database table within configurator and then use configurator to create table in database.

There are also other built in configuring tools – automatic verification of configuration, its graphical overview and also logging of running application with definable level of tracked details. Repeating tasks are facilitated using software wizards.

Synchronous and asynchronous computations

Synchronous and asynchronous processing of tasks can be provided within one integrated application implemented using OpcDbGateway. Every from them is fulfilled within own thread (Figure 5). Synchronous computations are run in synchronous controller (Figure 2) of the OpcDbGateway runtime application.

A synchronous processing goal is providing an environment to execute quasi parallel algorithms alike as e.g. in PLC used to control technological processes. PLC's has been initially created as substitution of control systems with hardwired contactors where has been a real parallelism in the functionality of different parts. If alike functionality has to be achieved in one processing unit, where in fact only serial processing is provided, then it is necessary to provide a transformation of the consistent input data set to the consistent output data set within defined computing period. If the period is short enough – negligible in relation to time constants of the controlled system, then the processing from the point of view of controlled system is parallel. Synchronous processing is executed in three steps:

The goal of the synchronous processing is providing an environment to execute quasi parallel algorithms.

- Reading of input data
- Computations
- An output data update

Of course, different internal data are used in computations but their transformations are not relevant for the controlled system – external connected devices. The OpcDbGateway in synchronous mode reads input data from different devices and databases. It processes them within defined period and put them to process databases or sends them to external devices. This type of processing is suitable not only as substitution of relay networks and digital integrated circuits but also to control continuous systems including multi parameter systems with time constants substantially higher as processing period. It can be used e.g. for simulation of dynamical systems or reading of interval consumption data from energy consumption meters. As already mentioned, particularity of synchronous computing is reading of complete input data set at the beginning of the processing cycle and the writing of complete output set at the cycle end. Integrator has simplified work when uses synchronous processing because a data reading and update from/to external devices and databases must not be explicitly configured.

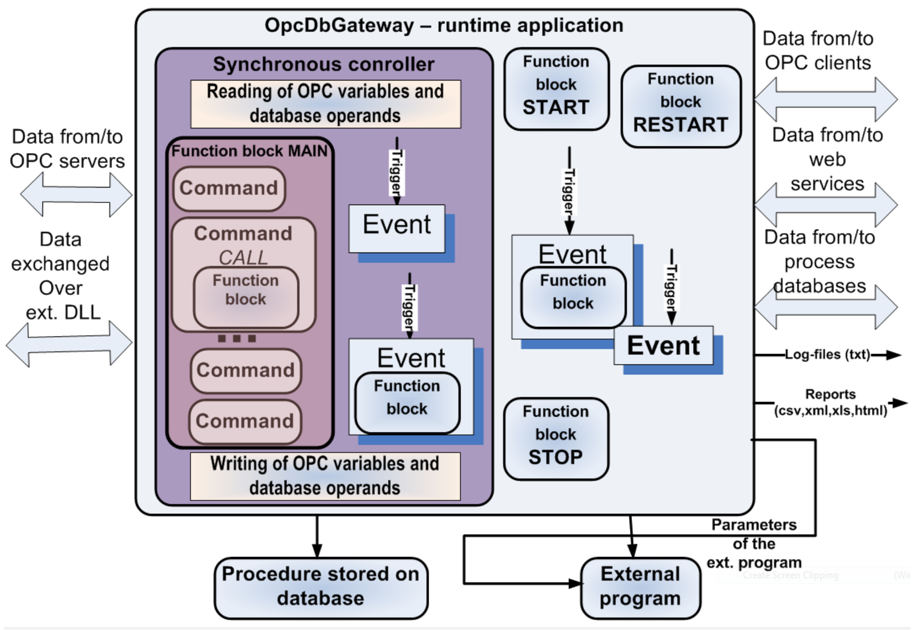
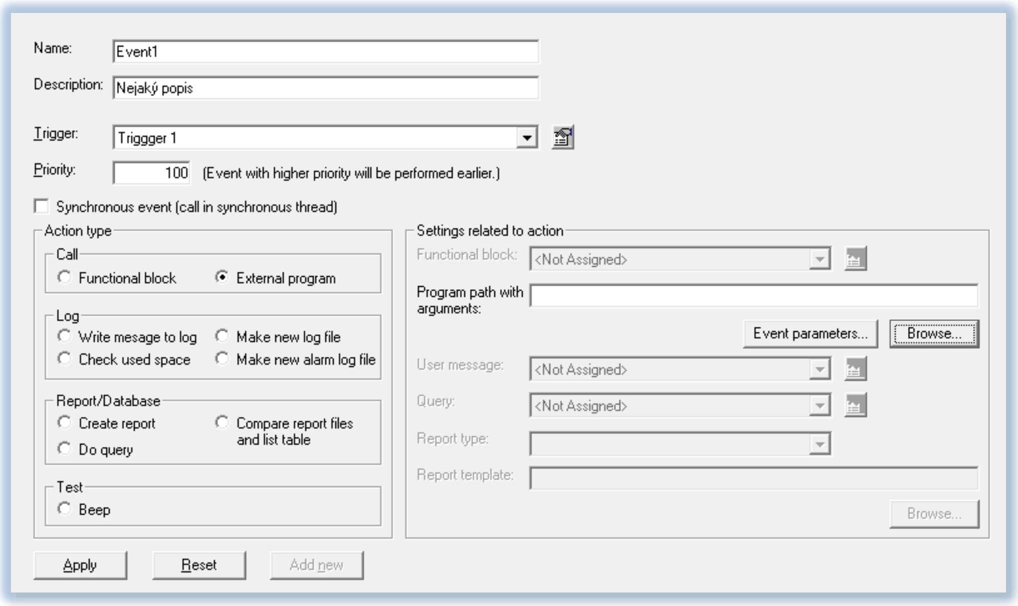


Figure 2 Integrator's model of the OpcDbGateway

But, he must understand this functionality to reconsider if it is suitable for a concrete application or not.

Triggers and events

Processed tasks in OpcDbGateway are entitled events. Every event (Figure 3) is started by a trigger. One trigger can start one or more events. Events have defined priority.



Name: Event1
Description: Nejaký popis
Trigger: Triggger 1
Priority: 100 (Event with higher priority will be performed earlier.)
☐ Synchronous event (call in synchronous thread)

Action type
- Call: ○ Functional block [✓ ● External program]
- Log: ○ Write mesage to log ○ Make new log file ○ Check used space ○ Make new alarm log file
- Report/Database: ○ Create report ○ Compare report files and list table ○ Do query
- Test: ○ Beep

Settings related to action
- Functional block: <Not Assigned>
- Program path with arguments:
- User message: <Not Assigned>
- Query: <Not Assigned>
- Report type:
- Report template:

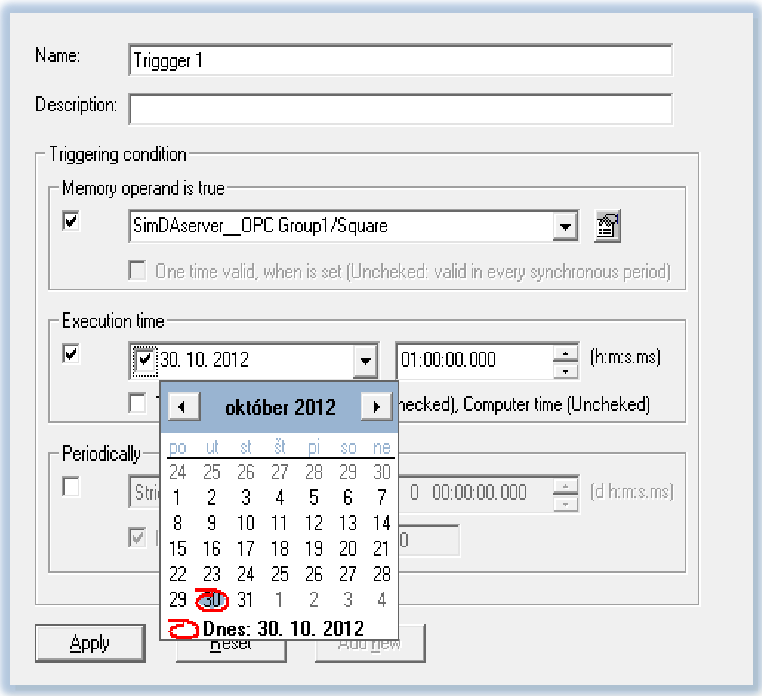
Figure 3 Dialog to configure events

Incurred event is put to one of two queues – synchronous and asynchronous. They are used by synchronous and asynchronous processing. The synchronous events are processed according to their priority in synchronous queue within the synchronous thread and the asynchronous according to their priority in asynchronous queue are processed within asynchronous processing thread. Executed events are immediately removed from queues. Switching of threads is done evenly. Consequently, event from asynchronous queue can be executed sooner than event in synchronous queue and vice versa.

Every trigger can contain one or two execution conditions: time or value of a memory operand or both (Figure 4). The triggers with condition time can be of one shot type or cyclical. It is possible (in dependency on a trigger type) to define start up time and number of cycles for the cyclic triggers.

A trigger activation used for putting of events to queues is executed in a separate thread (Figure 5). Evaluation of memory operands used in triggers with the only condition value is executed at the end of the synchronous cycle. Values of memory operands used in triggers with condition "time and value" are evaluated in the triggers time. When the trigger which starts an event executed from asynchronous queue is activated, the event is put immediately to the asynchronous queue according to its priority and evenly, if the priority (in the asynchronous queue) would be the highest, it can be executed immediately and removed from the queue. Synchronous queues are in fact two – preparation and execution queue. Trigger activation causes that an event is put to the preparation synchronous queue. At the beginning of the new synchronous cycle, the preparation and executing queue swap their tasks. It has following consequences:

- The event ordered in the synchronous queue is not changed within one synchronous period
- As the queues (preparing and executive) are swapping, the not executed events from previous period remain to execute in the next period. A long-time growing of events number in the synchronous queue is signalling that there is a problem in the configuration or in available computing resources. Integrator can watch the number of events in queue and eventually to solve the problem.


Name: Triggger 1

Description:

Triggering condition

Memory operand is true

✓ SimDAserver__OPC Group1/Square

☐ One time valid, when is set (Uncheked: valid in every synchronous period)

Execution time

✓ ✓ 30. 10. 2012 | 01:00:00.000 (h:m:s.ms)

☐ ecked), Computer time (Uncheked)

október 2012

po	ut	st	št	pi	so	ne
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Dnes: 30. 10. 2012

Periodically

☐ Stri | 0 00:00:00.000 (d h:m:s.ms)

✓ | 0

Apply | Reset | Add new

Figure 4 Dialog to configure triggers

Events affect not only the configurable computing functionality (*call function block*) but also directly provide configurable activation of the external (*call ext. program*), executing SQL queries on process databases (*do query, create report, sync. generated reports table*), generating of log-files and writing of configurable user messages to them (*make new alarm log file, make new log, write message to log*).

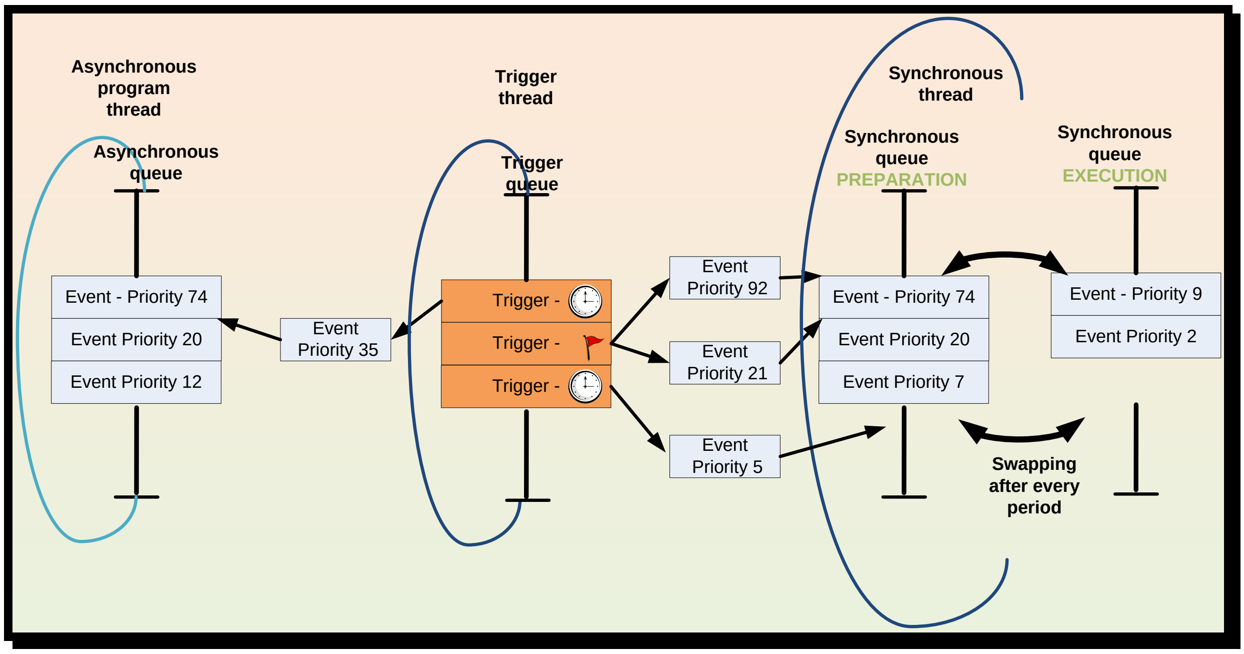


Figure 5 Threads for synchronous and asynchronous processing

Function blocks and commands

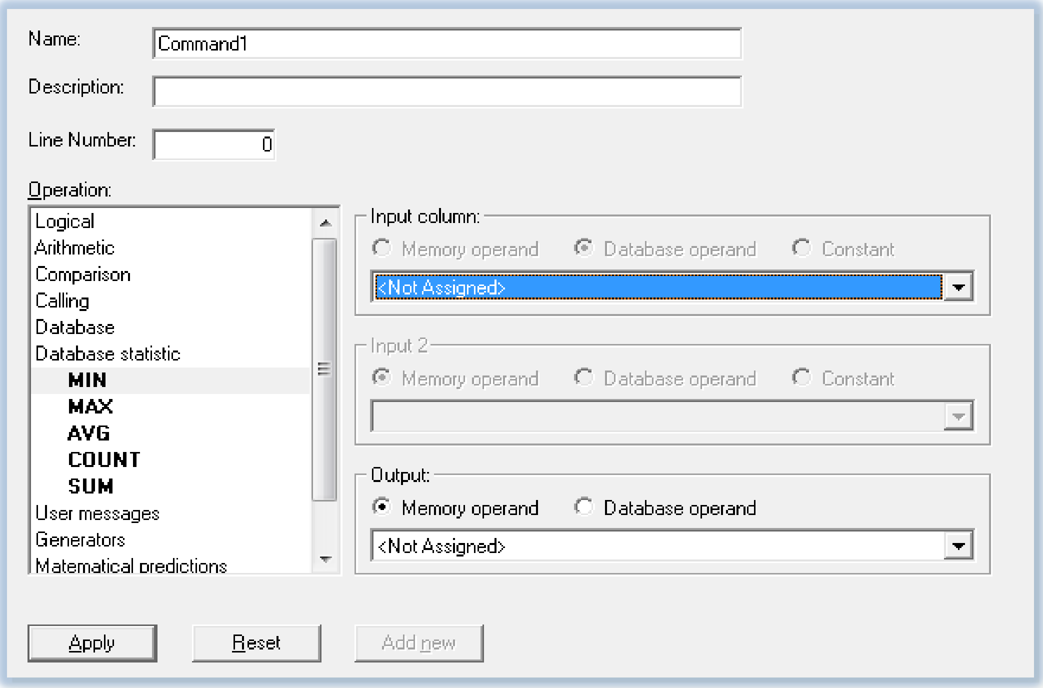
Data processing can be structured in function blocks. Configurable commands are executed within function blocks (Figure 8). Commands can have maximally one output parameter and two input parameters (Figure 7).

The commands sequence is defined by their numbering (Figure 8).. By a functional block execution, commands are executed in upward order. There can be gaps in the numbering. The command *CALL* enables calling of another function block. As an input parameter can be used for conditioned execution of this command, a branching of the program is enabled this way. Further explanations to commands will be done after explaining some other terms.

More demanding tasks can be programmed within OpcDbGateway using of high level programming languages and to integrate as new configurable functionalities to the OpcDbGateway as enhancing DLL's.

There are 4 predefined function blocks - MAIN, START, RESTART and STOP which are executed in every integrated application. These blocks do not always need contain some commands. The function block START is executed only one time when the application is started. It can be used to initialize the application. OpcDbGateway has implemented easy mechanism of the data persistency. If we want to use the data saved when the application is stopped later by restart of the application, the function block RESTART is called instead of START function block. The function block STOP is executed also only one time when the application is ending. Functional block MAIN is executed cyclically in the synchronous program thread.

Functional blocks can be called within events in the synchronous or asynchronous thread. Functional block MAIN is executed normally within synchronous thread as the last one - after executing all functional blocks called within events. It means, it behaves as it would have the lowest priority. But if the execution of all other function blocks did not enabled to execute the functional block MAIN, then it is called as the first one in the next cycle.



Name: Command1

Description:

Line Number: 0

Operation:

- Logical
- Arithmetic
- Comparison
- Calling
- Database
- Database statistic
  - MIN
  - MAX
  - AVG
  - COUNT
  - SUM
- User messages
- Generators
- Matematical predictions

Input column: ( ) Memory operand (●) Database operand ( ) Constant

<Not Assigned>

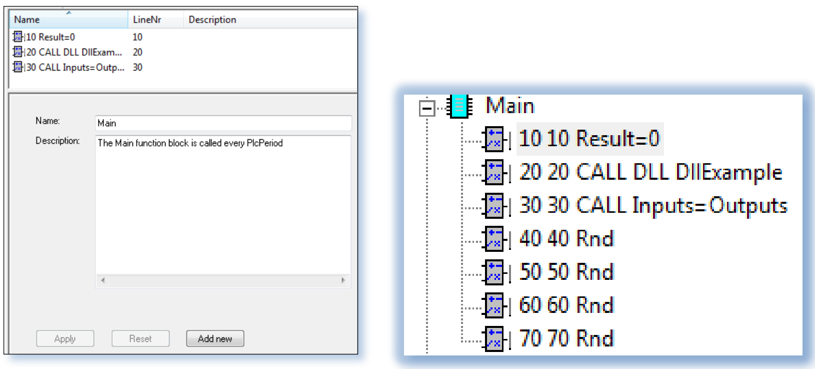
Input 2: (●) Memory operand ( ) Database operand ( ) Constant

Output: (●) Memory operand ( ) Database operand

<Not Assigned>

Figure 7 Dialogbox to configure commands

Programming constructions available using function blocks are not as snug as those provided by high level programming languages. But, they enable easily to configure (not to program) many typical tasks of the collecting, processing and intermediating of data between external devices. More demanding tasks can be still programmed within OpcDbGateway using of high level programming languages and to integrate to the OpcDbGateway as enhancing DLL's. The enhancing DLL can be used within a function block using the command *CALL DLL*.



Name	LineNr	Description
10 Result=0	10	
20 CALL DLL DllExam...	20	
30 CALL Inputs=Outp...	30	

Name: Main

Description: The Main function block is called every PlcPeriod

Main
- 10 10 Result=0
- 20 20 CALL DLL DllExample
- 30 30 CALL Inputs=Outputs
- 40 40 Rnd
- 50 50 Rnd
- 60 60 Rnd
- 70 70 Rnd

Figure 8 Configuring of function blocks

Memory and database operands

The memory operands area is the place where an information exchange between different external devices, databases and computation modules implemented as enhancing DLL's is executed (Figure 9).

Commands are executed using memory and database operands as parameters. Memory operands are placed to continuous memory area. An Integrator can choose the name of the memory operand and also its placement defined by numbering. It is possible to create folders when configuring the memory operands but the placement of the memory operand in the folder does not affect its placement in the memory operands area.

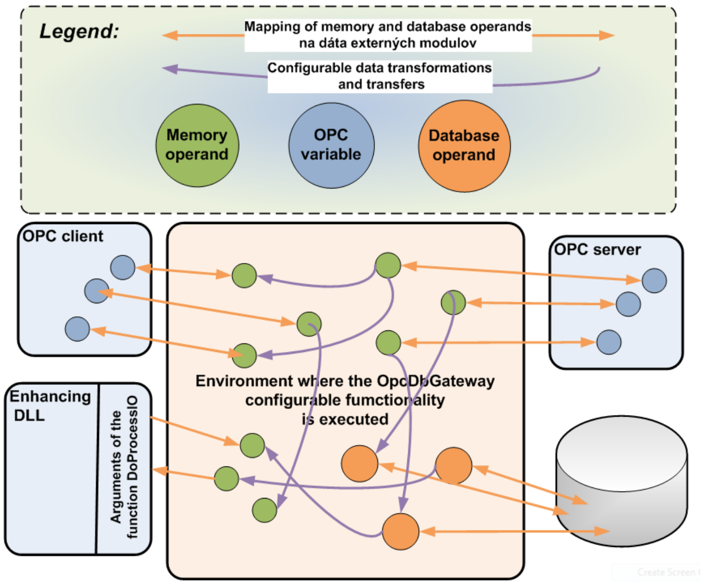

Legend:
Mapping of memory and database operands na dáta externých modulov
Configurable data transformations and transfers
Memory operand
OPC variable
Database operand
OPC client
OPC server
Enhancing DLL
Arguments of the function DoProcessIO
Environment where the OpcDbGateway configurable fumctionality is executed

Figure 9 Data exchange between external devices, databases and enhancing DLL's over memory and database operands

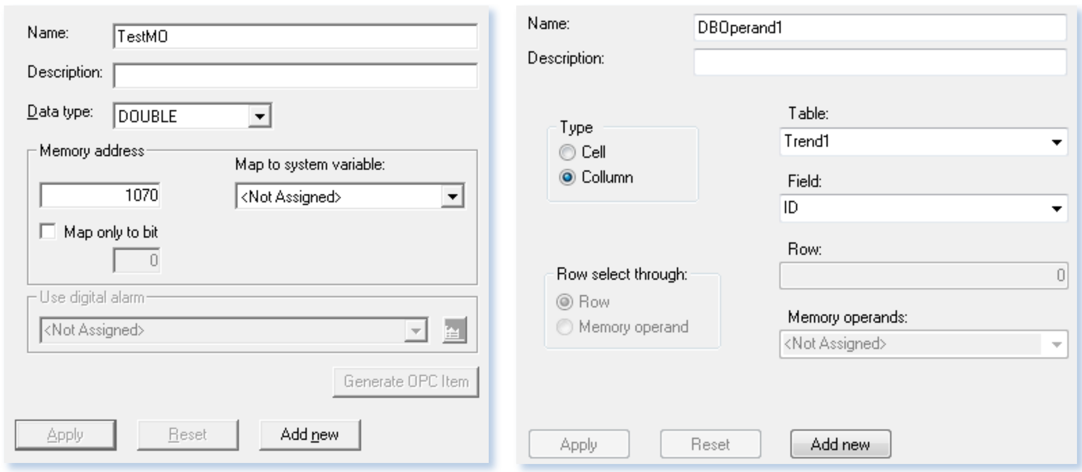
Data from external devices are mapped to the memory operands area. By using of the OpcDbGateway configurable functionality, we work with memory operands and provide processing and data exchange between external devices, databases, program modules and the computing machine of the OpcDbGateway itself.

The memory operands area is the place where an information exchange between different external devices, databases and computation modules implemented as enhancing DLL's is executed.

There is one special area of the memory operands with uniquely determined using as interface between the OpcDbGateway computation machine and the integrated application – the system operands area. Using it, the application can not only gain information about computing machine but also affect its functionality.

It is possible to choose a data type (Figure 10) of the memory operand. This way, the necessity of the implicit data type conversions is suppressed to achieve higher performance. Following data types can be used: *bool, byte, currency, date, double, dword, float, integer, long, short, string, word*. If the *bool* data type is chosen than a memory operand can be defined as one bit of the 16-bit word which is normally used by the data type *bool* to transfer information.

Memory operands can be used as sources to generate, or as source to quit an alarm. Because of this an alarm can be associated with a memory operand.


Name: TestMO
Description:
Data type: DOUBLE
Memory address: 1070
Map to system variable: <Not Assigned>
Map only to bit: 0
Use digital alarm: <Not Assigned>

Name: DBOperand1
Description:
Type: Cell [✓ Collumn]
Table: Trend1
Field: ID
Row: 0
Row select through: [✓ Row] Memory operand
Memory operands: <Not Assigned>

Figure 10 Dialogs to configure memory and database operands

Database operands and cooperation of OpcDbGateway with databases

Database operands are one of tools of the OpcDbGateway configurable functionality.
They are used to map a cell or column in the defined table of the chosen process database.
In some sense it can have behaviour of an indexed variable. As the index, a memory operand can be used. Changing value of this operand, a cell in the database table approached using a database operand is changed.

Database operands enable configurable data transfers between memory operands and themselves. There is e.g. the possibility to copy a value from a memory operand to the database operand. They can also be used as an argument in the configurable commands (Figure 11).

Database operands are important part of the synchronous data processing in OpcDbGateway. They are read in every synchronous cycle from database, and, after execution of related computations, their values are saved automatically to the process databases. It is up to integrator to decide if this kind of work within application is effective or not. Too often access to the database (in every synchronous cycle) can cause problems. In such case, it is more convenient using of asynchronous access to the database only in the time when it is necessary. There are some commands for this purpose using memory operands to access to the database. Very flexible approach is using of configurable SQL query, where memory operands, whose content can be changed in runtime, are used as parameters. It is also possible to start a database stored procedure using configurable SQL query.

As some commands use database operands as argument, then, if we would like to use them, we cannot keep off from implicit synchronous actualization of the database operands.
But, mostly also these commands, we can substitute using configurable SQL queries. The most flexible way of working with databases in OpcDbGateway is to program application specific database functionality within enhancing DLL. Of course, we pay for that by higher work expenditure.

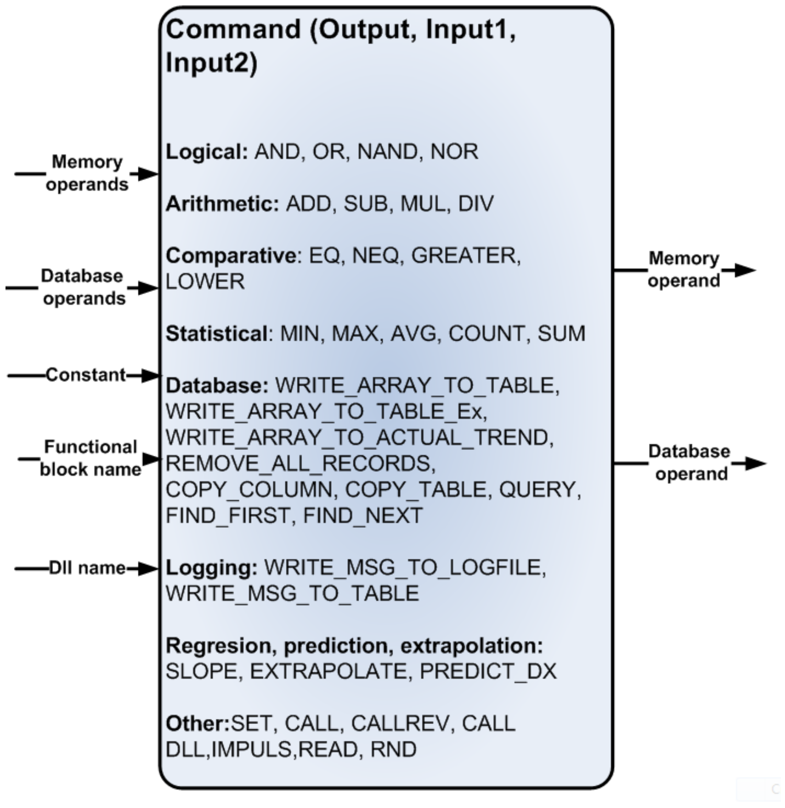

Command (Output, Input1, Input2)
Memory operands
Database operands
Constant
Functional block name
Dll name
Logical: AND, OR, NAND, NOR
Arithmetic: ADD, SUB, MUL, DIV
Comparative: EQ, NEQ, GREATER, LOWER
Statistical: MIN, MAX, AVG, COUNT, SUM
Database: WRITE_ARRAY_TO_TABLE, WRITE_ARRAY_TO_TABLE_Ex, WRITE_ARRAY_TO_ACTUAL_TREND, REMOVE_ALL_RECORDS, COPY_COLUMN, COPY_TABLE, QUERY, FIND_FIRST, FIND_NEXT
Logging: WRITE_MSG_TO_LOGFILE, WRITE_MSG_TO_TABLE
Regresion, prediction, extrapolation: SLOPE, EXTRAPOLATE, PREDICT_DX
Other:SET, CALL, CALLREV, CALL DLL,IMPULS,READ, RND
Memory operand
Database operand

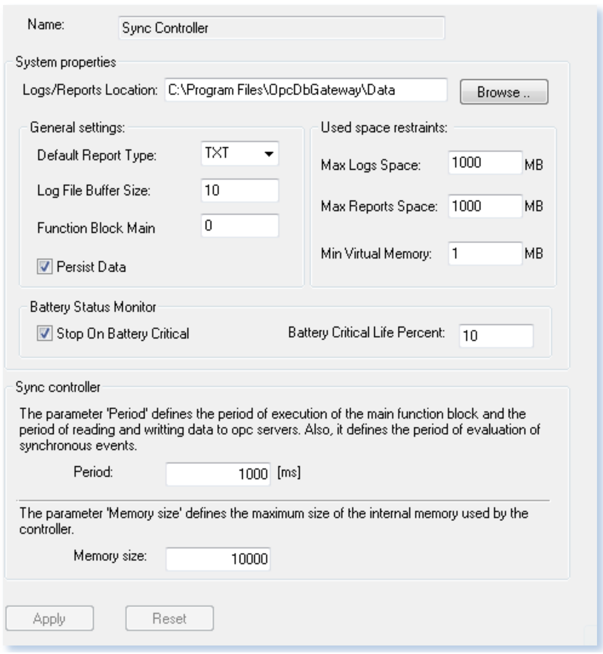
Figure 11 Configurable commands - overview

OpcDbGateway runtime functionality

Runtime application of the OpcDbGateway is implemented as a Windows program without own user interface. The runtime application is actually an OPC server providing access to memory operands. It can be started either by an OPC client application or as a Windows NT service. One or more OPC client applications can connect the OPC server of the OpcDbGateway. The configuration application of OpcDbGateway also contains the OPC client and so can be well used to start runtime application and to debug integrated application. Using it, system variables can be affected with impact on running of OpcDbGateway.

Synchronous computations are executed by the synchronous controller which can be configured using dialog box (Figure 12). The cycle period of that can be configured in milliseconds.

OpcDbGateway runtime contains **own OPC client** (Figure 13) enabling communication with external devices over external OPC servers. OPC items from those OPC servers can be mapped on memory operands and through them to be accessible for internal OPC server, other external OPC servers or for enhancing DLL's. There can be defined different requirements on updating of values from external OPC server.



Name: Sync Controller

System properties

Logs/Reports Location: C:\Program Files\OpcDbGateway\Data

General settings:

Default Report Type: TXT

Log File Buffer Size: 10

Function Block Main: 0

[✓] Persist Data

Used space restraints:

Max Logs Space: 1000 MB

Max Reports Space: 1000 MB

Min Virtual Memory: 1 MB

Battery Status Monitor

[✓] Stop On Battery Critical

Battery Critical Life Percent: 10

Sync controller

The parameter 'Period' defines the period of execution of the main function block and the period of reading and writting data to opc servers. Also, it defines the period of evaluation of synchronous events.

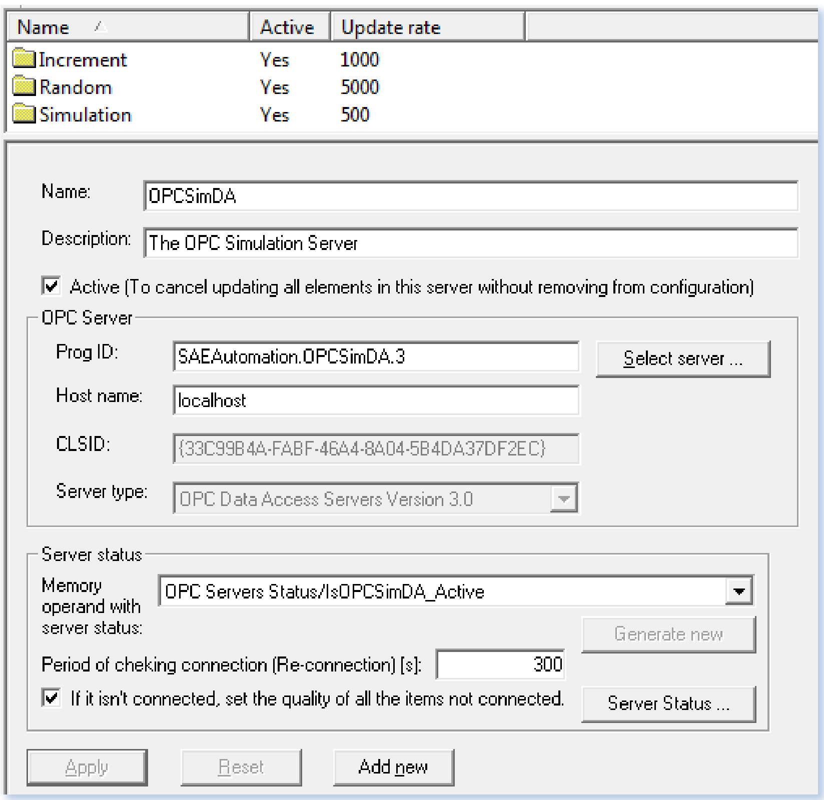
Period: 1000 [ms]

The parameter 'Memory size' defines the maximum size of the internal memory used by the controller.

Memory size: 10000

Figure 12 Configuring of the synchronous controller functionality

Because of this, within OPC client configuration can be



Name	Active	Update rate
Increment	Yes	1000
Random	Yes	5000
Simulation	Yes	500

Name: OPCSimDA

Description: The OPC Simulation Server

[✓] Active (To cancel updating all elements in this server without removing from configuration)

OPC Server

Prog ID: SAEAutomation.OPCSimDA.3

Host name: localhost

CLSID: {33C99B4A-FABF-46A4-8A04-5B4DA37DF2EC}

Server type: OPC Data Access Servers Version 3.0

Server status

Memory operand with server status: OPC Servers Status/IsOPCSimDA_Active

Period of cheking connection (Re-connection) [s]: 300

[✓] If it isn't connected, set the quality of all the items not connected.

Figure 13 Configuring of internal OPC client to access external OPC server

defined OPC groups with specific parameters as Update Rate, Dead band, Keep Alive functionality.

Within every synchronous controller cycle, OPC items from external OPC servers are automatically read and mapped memory operands are actualised.

Multitude of the OPC items which have to be updated in one cycle can be very high. Also those variables are updated in this cycle, by which none change has happened. There are two possibilities how to update mapped memory operands in every cycle. Either already mentioned synchronous reading of OPC items from external OPC server, or update of the mapped memory operands according to the cache of the internal OPC client. When using synchronous reading from external OPC server we have again two possibilities – either reading from the external OPC server cache or directly from an external device. Which of these methods is more convenient depends on the type of the device and on the type of communication protocol used between them and the external OPC server.

Reading from the OPC client cache will be mostly much more effective. OPC server notifies the internal OPC client about every OPC item change. The maximal frequency of these notifications can be set independently from the period of the synchronous controller cycle.

Many different OPC servers can be connected to the internal OPC client. It can easily happen that an external device is switched of and its OPC server is not more functioning. Such case may not affect functionality of the whole integrated application. Because of this, it is possible to parameterize the period of attempts of the internal OPC client to reconnect to the disconnected OPC server.

OPC client and internal OPC Server use own program threads which are basically independent to above mentioned three program threads – synchronous, asynchronous and timer thread. A way of the reading of data from memory operands by internal OPC server is, except of its possibilities, given mainly by parameterization of external OPC client applications.

Enhancing DLL modules

Enhancing DLL's enable substantially enhance functionality of the OpcDbGateway. In fact, using them, almost all functionality of an integrated application can be programmed within enhancing DLL instead of configuring it using configurable commands. To be able to choose which enhancing Dll's have to be used within an integrated application using the OpcDbGateway configuring application, every enhancing DLL has to have a few API functions to cooperate with configurator. Of course, there have to be also API functions providing cooperation with OpcDbGateway runtime.

There are two operating modes:

1. Calling functionality implemented in enhancing Dll's on demand – using configurable command *CALL DLL*
2. Continuous running of the enhancing Dll's functionality

In both modes, a data exchange between enhancing Dll and the OpcDbGateway runtime is provided over interface provided by memory operands. The first mode can be used e.g. to execute fast and complex computations which is easier to implement by programming as by configuring. The second mode is useful e.g. for communication drivers which have to be able react on notifications from external devices.

Sometimes it can be necessary to parameterize functionality of the enhancing DLL. It can be done either by standard functionality provided by configurator or special by enhancing Dll for configurator. In the first case, parameters can be configured as constants and at the beginning (within function blocks START or RESTART) copied to the memory operands using for interfacing of runtime and external Dll. This way can provide configuring of own enhancing Dll's every user. In actual version, there is not a possibility to create also user's enhancing Dll's for configurator because it requires recompilation of the whole configuration application..

Using enhancing Dll's for both configurator and runtime has been implemented **DDE client driver for OpcDbGateway and SAEAUT UNIVERSAL OPC Server.** Enhancing Dll for configurator enables fast mapping of DDE items from a DDE server to memory operands and OPC items of the internal OPC server.

Built-in SCADA functionalities

The computing machine of the OpcDbGateway has a few built-in standard functionalities usually provided by SCADA as e.g. alarming system, generating, administration and printing of reports, watching of own application environment and used resources as battery or the UPS status, using of virtual memory, place on HD used by process databases, log-files and reports. To communicate with them, the OPC variables on internal OPC server and to them mapped memory operands are used. Using system variable is e.g. possible to stop or start functionality of the synchronous controller. Visualisation and other application cooperating with OpcDbGateway can use OPC DA, OPC XML DA, OPC UA standards. These communication possibilities can be enhanced also with other communication protocols using enhancing DLL's. This way, e.g. a communication over DDE can be implemented.

The computing machine of the OpcDbGateway has a few built-in standard functionalities usually provided by SCADA.

Alarming system for integrated applications using OpcDbGateway can be implemented by two ways using of OPC AE standard or by special alarming system provided by computing machine of the OpcDbGateway. This system enables generating of an alarm based on more alarm sources and also quitting of one alarm from more quitting sources. Alarm system has, except of its status automat, the built in table of the alarm sources statuses as well as the alarm-log table. As the source of the alarm as well as source of its quitting is memory operand, setting or quitting of alarm can be provided by from different external devices connected to the OpcDbGateway.

The actual trend functionality is provided within table in process database with defined number of lines where the newest record is put at the beginning of the table and the other are moved one place down. A visualisation application connected to this table can show one or more moving graphs using data from the table. (Of course sometimes it can be more effective caching values in a visualisation application instead of repeated reading all values from database. Database table can be used only to fill up the application cache by starting of the application.)

The historical trends functionality provides cyclical saving of one or more values from memory operands to the process database table with growing number of lines. It is possible to define also periodic backing up of the table. Configuring of this functionality supported by a wizard is very easy. Creation of all parts of the configuration as triggers, events, SQL queries is executed automatically. The functionality of creating historical trend is cyclically activated by trigger within asynchronous thread. It means, it does not require running of synchronous controller.

User messages and alarm messages system enables configuring of parameterizable messages in more languages. The language used in run time is chosen by system variable. Messages can contain maximally 5 variables which are read in runtime from parameterized memory operands. User messages can be configured to be sent also as SMS or short e-mail.

Logging of the computing machine activities with configurable details is in English language. Information, warning and error messages are well distinguishable each other.

Reporting functionality of the OpcDbGateway is based on events calling queries on databases. Reports can be created in different formats – txt, csv, html, xls, snv. Formatting of reports can be defined by templates.

Methodology of the integrated application design

There is multitude of application types which can be integrated using OpcDbGateway. Very generally, to design them, next steps have to be executed:

1. Application goals definition.
2. Choosing if the application functionality will be based preferably on configuring (more functionality implemented using configurable commands) or on programming (more functionality provided by enhancing Dll's).
3. Choosing of external devices, programs databases and enhancing DLL's from which data will be processed.
4. External data model creation – interfaces between external entities and OpcDbGateway.
5. Partitioning of tasks on synchronous and asynchronous.
6. Function model design – function blocks.
7. Design of usage of the standard system components (alarms, historical trends...)
8. Design of other functionality using function blocks.
9. An internal data model design – memory and database operands.
10. Design of the stored database procedures.
11. Choosing a way how to start OpcDbGateway – by a OPC client application or as a Windows NT service

Conclusions

Set of features offered by OpcDbGateway to integrate applications is really rich. It enables processing of large amount of data from external devices with the synchronous controller cycle within periods from hundreds milliseconds to a few seconds and also activating of external enhancing DLL modules, external programs and scripts and stored procedure with lower periods and fast computing activities.

The possibility to work with more databases and easily map database tables to OpcDbGateway enables to use it for integration of enterprise applications where data from production devices have to be used together with data from warehouse system, OEE, ERP and other systems.

In addition to standard OPC interfaces based on DCOM technology, it enables using of web services according to the OPC XML DA or OPC UA standards or within enhancing DLL’s. It is possible to use it within large SOA (on services oriented architecture) systems. Usage of OpcDbGateway is effective also in small applications where e.g. communication between two external devices or between device and database has to be provided. It enables connect to the devices using DDE servers or to different applications with built-in DDE server.

An advantage of this product is the ability to augment its configurable functionality. Because of this, the integrator companies which use it can create an integrator’s software package accommodated to their specific needs.